Artisan Technology Group

Artisan Technology Group is your source for quality new and certified-used/pre-owned equipment

- FAST SHIPPING AND DELIVERY
- TENS OF THOUSANDS OF IN-STOCK ITEMS
- EQUIPMENT DEMOS
- HUNDREDS OF MANUFACTURERS SUPPORTED
- LEASING/MONTHLY RENTALS
- ITAR CERTIFIED SECURE ASSET SOLUTIONS

SERVICE CENTER REPAIRS
Experienced engineers and technicians on staff at our full-service, in-house repair center

WE BUY USED EQUIPMENT
Sell your excess, underutilized, and idle used equipment
We also offer credit for buy-backs and trade-ins
www.artisantg.com/WeBuyEquipment ↗

InstraView™ **REMOTE INSPECTION**
Remotely inspect equipment before purchasing with our interactive website at ***www.instraview.com*** ↗

LOOKING FOR MORE INFORMATION?
Visit us on the web at ***www.artisantg.com*** ↗ for more information on price quotations, drivers, technical specifications, manuals, and documentation

Contact us: (888) 88-SOURCE | sales@artisantg.com | www.artisantg.com

MDR

MIDAS Data Recorder

User Guide

Doc version 1.4.7, 27 November 2001

for software from version 1.4.4

VMETRO

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The information in this document is subject to change without notice and should not be construed as a commitment by VMETRO. While reasonable precautions have been taken, VMETRO assumes no responsibility for any errors that may appear in this document.

VMETRO

Copyright ©1998-2001 VMETRO
This document may not be furnished or disclosed to any third party and may not be copied or reproduced in any form, electronic, mechanical, or otherwise, in whole or in part, without the prior written consent of VMETRO Inc. (Houston, TX, USA) or VMETRO asa (Oslo, Norway).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Warranty

VMETRO products are warranted against defective materials and workmanship within the warranty period of 1 (one) year from the date of invoice. Within the warranty period, VMETRO will, free of charge, repair or replace any defective unit covered by this warranty, shipping prepaid. A Return Authorization Code should be obtained from VMETRO prior to the return of any defective product. With any returned product, a written description of the nature of the malfunction should be enclosed. The product must be shipped in its original shipping container or similar packaging with sufficient mechanical and electrical protection in order to maintain warranty.

This warranty assumes normal use. Products subjected to unreasonably rough handling, negligence, abnormal voltages, abrasion, unauthorized parts replacement and repairs, or theft are not covered by this warranty and will, if possible, be repaired for time and material charges in effect at the time of repair.

VMETRO's warranty is limited to the repair or replacement policy described above and neither VMETRO nor its agent shall be responsible for consequential or special damages related to the use of their products.

Limited Liability

VMETRO does not assume any liability arising out of the application or use of any product described herein; neither does it convey any license under its patent rights nor the rights of others. VMETRO products are not designed, intended, or authorized for use as components in systems intended to support or sustain life, or for any application in which failure of the VMETRO product could create a situation where personal injury or death may occur. Should Buyer purchase or use VMETRO products for any such unintended or unauthorized application, Buyer shall indemnify and hold VMETRO and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that VMETRO was negligent regarding the design or manufacture of the part.

USA/Canada:

VMETRO, Inc. 1880 Dairy Ashford, Suite 400,
Houston TX 77077, U.S.A.
Phone: (281) 584-0728
Fax: (281) 584-9034
e-mail: info@vmetro.com

Europe/Asia:

VMETRO asa
Brynsveien 5
0667 OSLO, Norway
Phone: +47 22 10 60 90
Fax: +47 22 10 62 02
e-mail: info@vmetro.no

www.vmetro.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Conventions used in this document

The following section describes conventions used in this document.

Symbols	Meaning
	The STOP symbol indicates a section of critical importance. Overlooking this information may cause damage to the MDR and/or other equipment.
	Indicates important, but not crucial information. Still, you should take notice if you want to use all capabilities built into the MDR.

IEC Prefixes for binary multiples

Symbol	Name	Origin	Derivation
Ki	Kibi	Kilobinary	kilo
Mi	Mebi	Megabinary	mega
Gi	Gibi	Gigabinary	giga

Example:
1 Kibit = 1024 bit
1 MiB = 1 048 576 bytes

Related Documentation

- MDR-220 Quick Guide
- MDR-250 Quick Guide
- MDR Release Notes

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

1 Introduction 1

1.1 Features & Performance 3
1.2 Hardware Components 4
1.3 Software Components 5
1.4 Host System Requirements 5
1.5 Host Computer Software (SDK CD-ROM) 5
1.6 MDR Preloaded Software (MDR Firmware) 6
1.7 Data Storage Format 6
1.8 MDR Swinging Buffers 7
1.9 Data Readback 8
1.10 Supported PMCs 9

2 Hardware Installation 10

2.1 Handling Precautions 10
2.2 MDR Module Board Layout 11
2.3 MDR Board Switch and Jumper Locations 12
2.4 Installing the MDR Module into the VME Chassis 13
2.5 Cable Installation 13
2.6 Power Consumption 13
2.7 Hardware Configuration 13

3 Standalone MDR 14

3.1 MDR Software 14
3.2 VxWorks Boot Parameters for the MDR board 14
3.3 Configuring the MDR for Standalone Operation 15
3.4 Connecting to the MDR Server from the MDR Shell 16
3.5 Converting from Network to Standalone MDR 17

4 Host Network-Based MDR 18

4.1 MDR Host Software 18
4.2 Host MDR Shell and API Installation 19
4.3 Configuring the MDR Shell 20
4.4 Configuring the MDR for Ethernet Operation 21
4.5 Configuring the MDR for Shared Memory Network Operation 22
4.6 Connecting to the MDR Server from the MDR Shell 24

5 Host Direct-Connection MDR 25

5.1 Configuring the MDR Host Software for Direct Connection 25
5.2 Installing the Fibre Channel driver for Windows or Solaris 26
5.3 Connecting the MDR Shell to the disk subsystem 29

6 Host Network and Direct Connection MDR 30

6.1 Configuring the MDR for Network & Direct Connection 30

7 MDR Shell Commands 31

MDR Shell Command Description 32

8 MDR API Functions 45

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.1 Header Files ... 45
8.2 Error Conditions ... 46
8.3 API Function Description ... 46
8.4 MDR API Example Programs ... 61
8.5 MDR API Limitation ... 62
8.6 Compilation Instructions ... 62
8.7 Externals Libraries ... 63

9 MDR Operations 64

9.1 Disk Groups ... 64
9.2 Transfer Operations ... 65
9.3 Recovery from power failure ... 69

10 Configuration Files 71

10.1 Summary Table ... 72
10.2 Item modification ... 73
10.3 MDR system configuration file (mdr.ini) ... 74
10.4 VxWorks configuration file (vxbsp.ini) ... 78
10.5 VxWorks boot parameters file (vxbp.txt) ... 80
10.6 MIDAS Monitor configuration file (mmon.ini) ... 81
10.7 Configuring an MDR Unit by Preloading the Boot Parameter Files ... 82

A Troubleshooting 85

B Error Messages 86

C Upgrading the MDR System Software 89

MDR Firmware (mdrvxst.hex) ... 89
Version Upgrade Matrix ... 89
Restoring Corrupt FLASH ... 92

D Supported PMCs Documentation 94

Fibre Channel Storage ... 94
FPDP - Digital I/O ... 97
Ethernet Host Interface ... 105
SCSI PMC Backup ... 107

E MDR Standard Configurations 110

F Miscellaneous Functionalities 111

MDR Status Block mapped to VMEbus ... 111
MDR Server boot status ... 112

G Glossary 115

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 Introduction

The MDR (MIDAS Data Recorder) is a family of ready-to-run VMEbus-based subsystems that offer up to Terabytes of real-time permanent recording or playback of high-speed digital data. Continuous recording at rates up to 90 MB/s can be performed for minutes or even hours, limited only by the capacity of the disk storage system. The MDR is typically used to record raw sensor data directly from the data path between A/D converters and DSP processors in systems such as radars, sonars etc. Versions of the product are also capable of doing fixed-rate playback of recorded or preloaded data at speeds up to 72 MB/s, to feed data into a sensor data path for sensor emulation, system testing or verification purposes.

The MDR system hardware consists of four primary components:

1) Pre-programmed, self-contained VMEbus module
    - MDR-220 with two PMC sites
    - MDR-250 with five PMC sites
2) PMC modules installed into sites onto the MDR boards
    - DPIO-xI - data input interface to convert 32-bit parallel FPDP to PCI
    - DPIO-xO - data output interface to convert PCI to 32-bit parallel FPDP
    - VMFC-2100 - PMC Fiber Channel Adapter
    - Ethernet host interface supporting TCP/IP over 10/100 Mbits/s
3) Mass storage system based on 3rd-party Fibre Channel disk arrays.
    - RAID - Redundant Array of Inexpensive Disks
    - JBOD - Just a Bunch of Disks
4) PCI Module plugged directly into Host Computer
    - VMFC-2100P - (or VMFC-2200P) Fiber Channel Disk Interface Adapter

All main components of an MDR system are based on industry standards (VME, PMC/PCI, Fibre Channel) and commercial off-the-shelf (COTS) modules, such as the VMETRO MIDAS board and selected PMC modules. The MDR board is intended to be integrated into a customer's VME chassis, typically together with 3rd-party ADC and DSP boards. DPIO boards are based on industry-standard FPDP parallel port for compatibility with 3rd-party A/D converters (ADCs) and DSPs, or variants of this interface using Differential TTL (RS422), PECL or LVDS signaling.

The MDR system software consists of a variety of components provided either as an SDK (Software Distribution Kit) CD-ROM for user installation onto a Host workstation or firmware preloaded into FLASH on the MDR board.

The SDK is to be installed onto a host computer and provides the following components.

- MDR Shell (MDR Client) Host-based module
- MDR API (C/C++)
- Device Drivers (i.e., fibre channel PMC, etc.)

The firmware applications preloaded in the MDR FLASH provide all functionality necessary to run the MDR system. This package includes firmware applications and necessary configuration files. The MDR resident applications are:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- MDR Server (can only be run on an MDR board)
- MDR Shell (MDR Client) Standalone module

Using the MDR software and firmware, the MDR system may be operated in either of four modes depending upon system requirements:

1. MDR Standalone - MDR-2x0 runs both MDR Server and Shell. No Host workstation required. Monitor with VT100 terminal.
2. Host Network-based - MDR Shell on Host workstation and MDR Server on MDR-2x0 board.
    - Using MDR-250 with Ethernet PMC
    - Using MDR-220 with Shared-Memory Network (SMN) to SBC
3. Host Direct-Connection - MDR software on Host workstation communicates via FC HBA to disk array. No MDR-2x0 board required.
4. Host Network and Direct Connection - combination of modes 2 and 3.

Except in Host Direct Connection, all the MDR data transfer occurs through the MDR Server application resident in the MDR-2x0 board. The MDR Server is necessary for data recording capability. In Direct Connection, the MDR Server is bypassed and no recording may take place, although files may be transferred between the workstation and the disk array. When both Network and Direct connections are present, the setting of the `UseHostFcAdapter` parameter in the Host mdr.ini file determines which path is used to access the disk array and which functions are active.

The MDR shell (MDR client) is run on the Host in the host-based mode and is run on the MDR-2x0 in the standalone mode. The MDR Server, running on the MDR board, normally performs the disk array management, although in Direct Connection, the disk array is accessed and can be managed directly using a Fiber Channel adapter in the host workstation.

Typically, in the system development phase, the MDR is controlled from a host computer using the C/C++ API or a text-based user interface (MDR Shell). The host may communicate with the MDR using standard TCP/IP over Ethernet or with a high-speed TCP/IP protocol (SMN) over VMEbus. The host can be one of four possible computer systems as detailed in Section 1.5: *Host System Requirements*.

In Standalone operation, a VT100 terminal or terminal emulator is attached to the MDR RS232 port. In this case both the MDR Server and the Client (MDR Shell) programs run in the MDR board. This configuration is often used for host-independent operation of the MDR, suitable for in-the-field recording missions.

In Direct Connection operation, a dedicated Fibre Channel link is established between the host computer and the disk array. This is done to speed up read back of recorded data by the host or to preload data for playback into a system. In this configuration, none of the MDR Server calls are used, with all data manipulation performed directly by the host computer. Full Disk Group functionality is only available in Direct Connection.
The MDR is available with standard commercial environmental specifications, with an operating temperature range of 0-50°C for the MDR board and 0-25°C for the disk storage system. A ruggedized version designed to withstand specified amounts of shock, vi-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

bration and extended temperature range for both the MDR board and the disk storage system is under development.

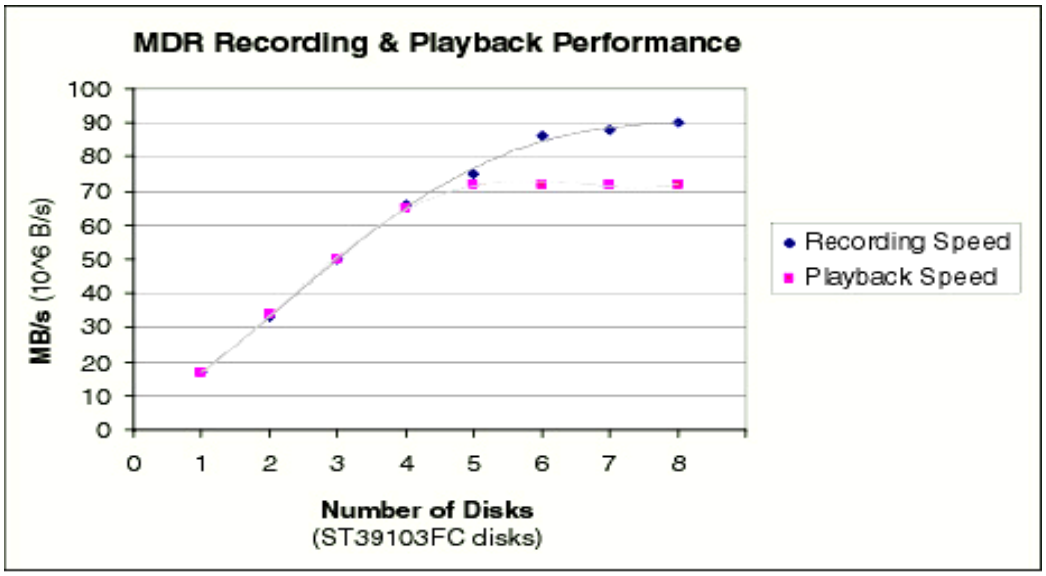
Note that it is not possible to write recordings in ASCII format with the MDR. The MDR only writes recordings in a binary format. The user must convert data accordingly.

1.1 Features & Performance

MDR subsystems incorporate the features leading to the associated benefits.

Features	Benefits
Up to 90 MB/s real-time permanent ADCs for recording for minutes or hours	Record raw sensor data directly from radar algorithm development, mission analysis
Status tags	Data recovery after power failure
FPDP TTL parallel interface	Compatibility with 3rd-party ADCs and DSPs
Fibre Channel disk interface	High speed, very scalable, convenient cabling
JBOD disk arrays	Very cost effective, high capacity, scalable
RAID disk arrays	Redundancy offers protection against disk failure
Controlled with User I/F or C++ API	Ready to run, or control from user application
Host access via Ethernet or VMEbus	Use workstation or an embedded VME host
Direct Fibre Channel host access to disks	Very fast readback of recorded data by host
User I/F accessible through RS232 port	Host-independent operation, ideal in the field
Optional playback at up to 72 MB/s	Sensor emulation, system testing or verification
Optional SCSI Tape Station (DLT/AIT-1/2)	Direct backup of recorded data

MDR Systems have been tested to provide the following performance.



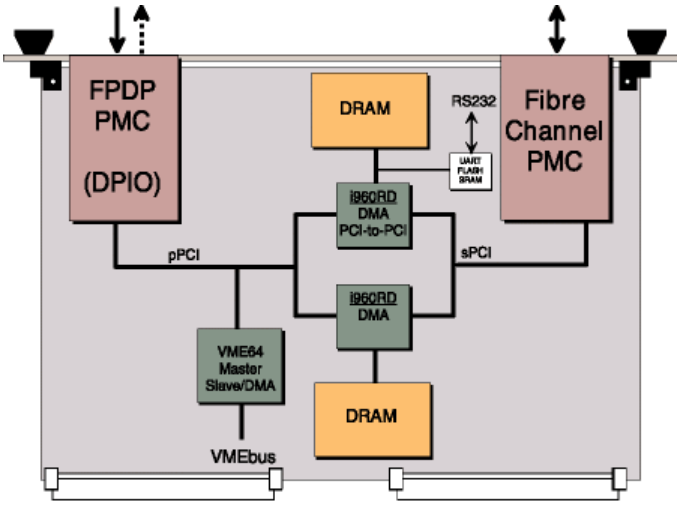
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 Hardware Components

The components available in the current MDR series contain powerful Twin i960 & Memory and support either two (2) or five (5) PMC Carriers (MDR-220 or MDR-250). The architecture couples each processor and memory array to a PMC I/O Sub-System, for maximum data throughput. These are MIDAS boards with the necessary software embedded in on-board FLASH memory to support the MDR PMC combinations.

1.2.1 MDR-220

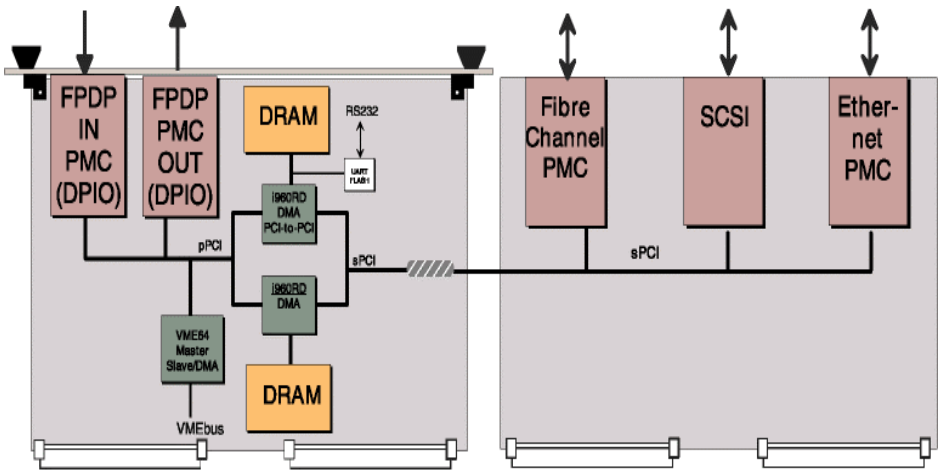
Two PMCs positions – one for FPDP (data source) and the other for the Fibre Channel connection to the disk array. Typically used for standalone or Shared Memory Network.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2.2 MDR-250

Five PMCs positions for more versatile I/O configurations. This configuration is the preferred method for providing Ethernet connection to a host computer, because an additional VME SBC is not required to provide the network gateway.



For further details, see *Appendix E: MDR Standard Configurations*.

1.3 Software Components

MDR-HOST-SW	MDR host software contains MDR Shell and API for host workstation (Only one required per project).
MDR-xx0	Base software run-time licenses included.

1.4 Host System Requirements

Host architectures:

- Sun SparcStation with Solaris 2.6 (or higher) and Ethernet interface,
- i386 (or higher) PC with Windows NT/2000 and Ethernet interface,
- MIDAS with VxWorks 5.4,
- Power PC SBC with VxWorks 5.3.1/5.4.

1.5 Host Computer Software (SDK CD-ROM)

The MDR can be connected to the host computer via the RS-232 port or via Ethernet (requires an Ethernet PMC on the MDR-250 base unit or VME SBC for network gateway). All the software needed to run the MDR system from any of the above hosts is contained in a CD-ROM. The Host MDR Shell relies on the mdr.ini file. The mdr.ini is located in the host/etc directory and must be modified with a text-only editor (i.e., Notepad) to set parameters such as the number of disks.

For further details on the MDR Host software, see Section 4: *MDR Host Software.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.6 MDR Preloaded Software (MDR Firmware)

The most significant software module in the MDR system is the MDR Server. The MDR Server processes all accesses to the disk array (except Direct Connection). The key configuration files used by the MDR Server are listed here.

- mmon.ini – must be present to run the MDR, seldom changes
- vxbsp.ini – user may need to edit this file to modify parameters
- vxbp.txt – contains VxWorks boot parameters. User will only edit when changing boot method (i.e., standalone, shared memory, Ethernet)
- mdr.ini – used to pass information about the physical configuration and options (i.e., MDR model, disk info, DPIO and FC parameters)

The MDR Shell is the user process used to request services from the MDR Server. Note that the disk group function (i.e., named groups) is not available when running the MDR Shell on the MDR board (available only in MDR on Host).

1.7 Data Storage Format

MDR relies on arrays of commercial disk drives as the permanent storage medium, and the very high recording performance is obtained by striping (interleaving) data across multiple disks. The typical storage system of an MDR has a storage capacity that ranges from tens of Gigabytes up to Terabytes, with disks configured as cost-effective JBOD (Just a Bunch Of Disks) units or as complete integrated RAID (Redundant Arrays of Independent Disks) units with built-in redundancy.

The disk units are connected to MDR with Fibre Channel Arbitrated Loop (FC-AL) operating at 1.062 Gbit/sec, using the SCSI protocol and command set to communicate with the disk system. Up to 125 disks units may be connected to the same physical loop. MDR supports Fibre Channel both with fiber optic or differential twinax copper cables, and hubs and switches can be incorporated in the storage system to obtain convenient cabling, easy reconfiguration, maximum flexibility and scalability.

The MDR storage system is organized as a *block device*: a block device divides the storage medium into a number of blocks, where a unique number addresses each block. The storage system is typically much larger than each recording, therefore the MDR implements functionality for storing up to 248 recordings. Each recording must be given the following attributes:

- Name
- Block number identifying the start of the recording
- Number of blocks specifying the size of the recording, or block number identifying the end of the recording.

The user must specify these parameters before a recording is started. The recording parameters of all recordings are stored in the recording table, which is located in a system-reserved partition of the storage device(s).

In order to achieve high speed and deterministic performance, the MDR implements a recording table instead of a traditional file system. This means that the user has direct

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

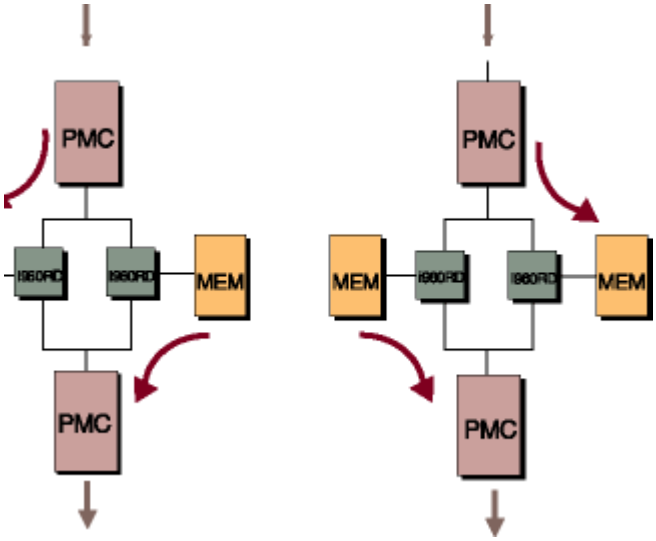
control over the start block and size parameters for each recording. An example of the recording table is shown after several recordings have been defined. Note that the prompt at the bottom always displays the name and size of the current recording.

```
Recorder Info:
  Total capacity          :  33.9 GiB  (  17365 blocks)
  Capacity used           : 510.0 MiB  (    255 blocks)
  Capacity available      :  33.4 GiB  (  17110 blocks)
  Largest recording size  :  33.4 GiB  (  17110 blocks)
  Recorder block size     :   2.0 MiB
  Number of recordings    :     2      (maximum : 895)

 RecNo       Blocks    UserSize      Name      MB/s Tag Stored       State
   1       0 -     4   10.0 MiB  Recording_1 28.5  No   Yes   Recorded
   2       5 -   254   00.0 MiB  Recording_2 39.9  No   Yes   Recorded / Current

Recording_2 (500.0 MiB) >
```

The hardware and software architecture of MDR is designed around a 32-bit data path. However, to facilitate data I/O widths less than 32-bit without losing bandwidth, various data packing and unpacking options are available in the data interface. This is a very useful feature when connecting to ADC and DAC boards, which often have sample widths of only 8 to 16 bits. For example, if a 10-bit ADC is used, the MDR can pack three 10-bit samples in a 32-bit word before it gets recorded, wasting only two bits rather than 22 if no packing were used. The packing options provided are 2x16-bit, 3x10-bit, 4x8-bit or 8x4-bits words packed in or unpacked from a 32-bit word.

1.8 MDR Swinging Buffers

The MDR base boards are equipped with two high-speed DRAM buffers which are utilized in a “swinging buffer” mode by DMA controllers in the input and output interfaces. This process ensures contention free input and output data flows through the MDR board, making a continuous recording performance of up to 90 MB/s possible.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.9 Data Readback

MDR provides several ways to read recorded data back from the disk storage system to a host system (or vice versa for preload in a playback application).

1.9.1 Copy to File on Host via network

Recorded data (or part of it) may be copied to one or several files on the host via the MDR network connection (Ethernet). The read back rate via Fast Ethernet is typically less than 1 MB/s.

1.9.2 Copy to File on Host via Fibre Channel

For the fastest possible read back of the data from the disk storage system, the RAID or JBOD disk storage system may be attached directly to the host, as shown in Chapter 5: *Host Direct Connection MDR*. This requires that the host is equipped with a Fibre Channel adapter (VMFC-2x00x-xx), and normally gives a read back speed of 40-80 MB/s, depending on the host system. In this case, the MDR board itself does not need to be present.

1.9.3 Simultaneous Recording and Host Read/Write of disks

Read or write of disk data by the host normally takes place “offline”, after a recording is completed. However, MDR also supports low-bandwidth host read or write traffic concurrently with recording. Since simultaneous host access will affect the “smooth” operation of the swinging buffer DMA controllers, this is associated with a certain performance degradation. For example, 500 KB/s host traffic may result in approx. 20% reduction in maximum recording speed.

1.9.4 Disk Array Physically Moved to a Different Host

The RAID or JBOD disk storage system may also be attached to a different host (equipped with a VMFC-2x00x-xx Fibre Channel adapter) than the one used during recording, i.e. the host and disk storage system is totally de-coupled from the MDR board itself. This can for example be used to physically move the disk array to a lab after the recording is done in a deployed system in the field. In a playback application, this can for example be used to preload data on the disk array before it is physically moved to the site where real-time playback will take place through the MDR board.

1.9.5 Read to Host Memory using the API

The MDR API provides a function for reading parts of a recording to a user memory buffer on the host system. This allows for post-processing of the recorded data by the host without first having to store data to a file.

1.9.6 SCSI Tape Station

For backup purposes, the recorded data may be transferred from the disk storage system onto DLT or AIT-1/2 tapes using a SCSI tape station attached to the optional SCSI controller. The transfer to tape is controlled by the MDR Shell/API.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.10 Supported PMCs

The PMCs currently supported by the MDR module are:

- VMFC-2100 Fibre Channel PMC
- DPIO FPDP PMC
- DE520 Ethernet PMC
- SC-PMC/875 SCSI PMC

1.10.1 Fibre Channel Storage

The permanent storage medium used by MDR is based on commercial hard disks using the FC-AL (Fibre Channel Arbitrated Loop) interface. MDR gets its disk recording performance from using an array of Fibre Channel disks such as **RAID** (Redundant Arrays of Inexpensive Disks) or **JBOD** (Just a Bunch Of Disks) units.

1.10.2 FPDP - Digital I/O

An MDR system normally interfaces to its data source with a digital I/O interface based on the industry standard FPDP (Front Panel Data Port) specification (Proprietary ports may also be accommodated; please consult factory). The FPDP interface is a 32-bit synchronous input/output parallel interface which is specified to operate at clock rates of 20 MHz (TTL) @ 32 bits, 25 MHz (PECL) @ 32 bits, and 40 MHz (PECL) @ 16 bits. The FPDP interface is used by a variety of third party vendors with products such as A/D converters, D/A converters and DSP boards.

1.10.3 Ethernet Host Interface

The MDR system utilizes an Ethernet PMC module to provide networking access to hosts like a PC or workstation. The Ethernet PMC modules supported are the Compaq DE520 and Rockwell Network Systems 2350 PMC FAST ETHERNET.

1.10.4 SCSI PMC Backup

The MDR uses the SCSI PMC module to provide connectivity to SCSI tape stations for backup purposes. The SCSI PMC current module supported is SC-PMC/875 from Concurrent Technologies.

For further information on the supported PMCs, see *Appendix D: Supported PMCs Documentation*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 Hardware Installation

2.1 Handling Precautions

The MDR hardware components are sensitive to static electricity and can be damaged by a static discharge. Always wear a grounded anti-static wrist strap and use grounded, static protected work surfaces when touching the hardware components. When the components are not in use, always keep them in the anti-static protective envelope.

2.1.1 Unpacking

All precautions described above must be taken when unpacking the MDR module from its shipping package. Verify that no damage has occurred in the shipment. Refer to the packing list and verify that all items are accounted for. (Items may vary according to the MDR-2x0 product purchased.)

Each shipment of an MDR board assembly comprises:

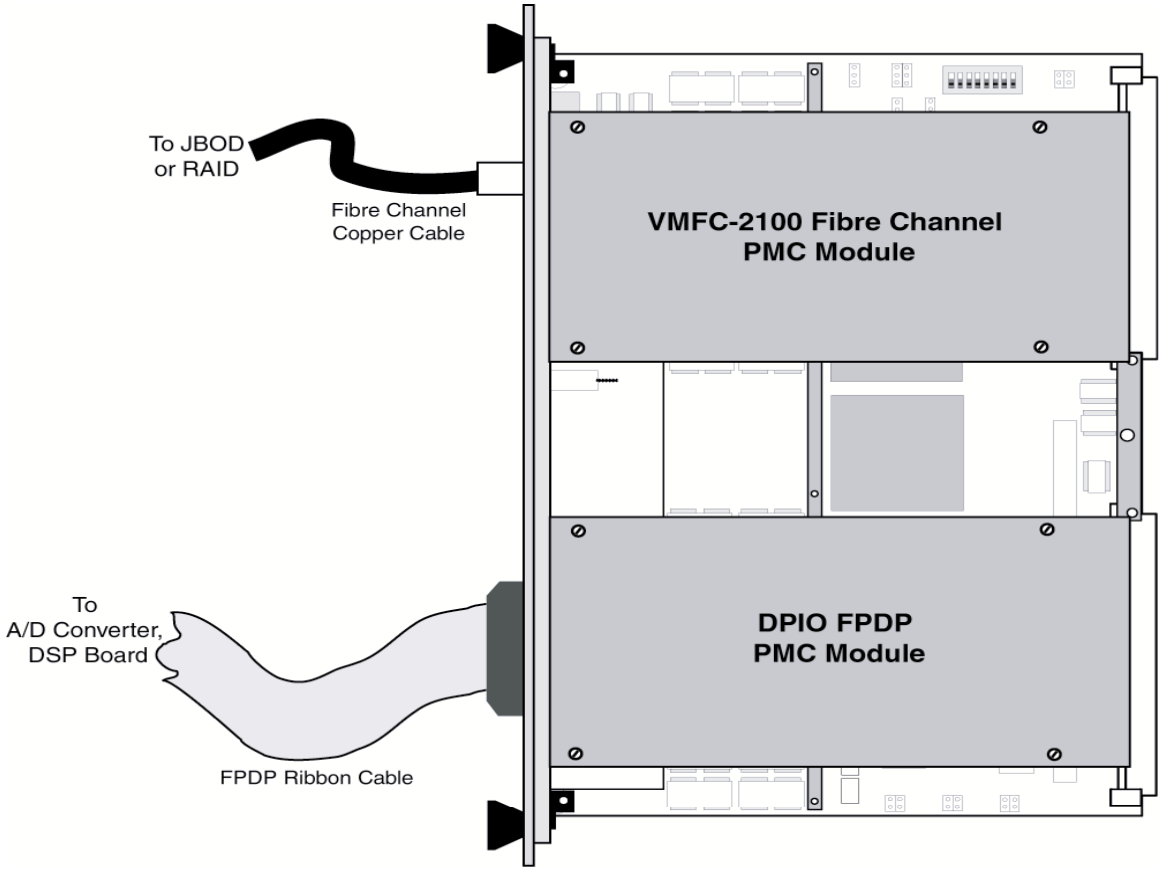
- 1 MDR Board
- PMC Module(s) preinstalled on MDR board and firmware in FLASH
- Cables
  - RS-232 cable
  - Fibre Channel cable, 3m / 10’
  - FPDP 80-pin Ribbon cable w/ KEL connectors, 1m / 3.3’
- Documentation - this User Guide- Quick Guide- MDR PMC Software License (one per PMC)

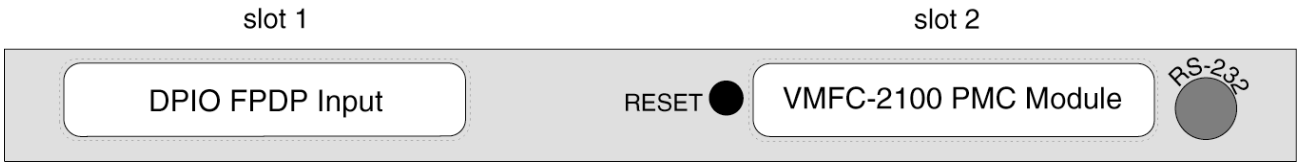
The MDR Host Software (part # MDR-HOST-SW) on CD-ROM is ordered separately.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

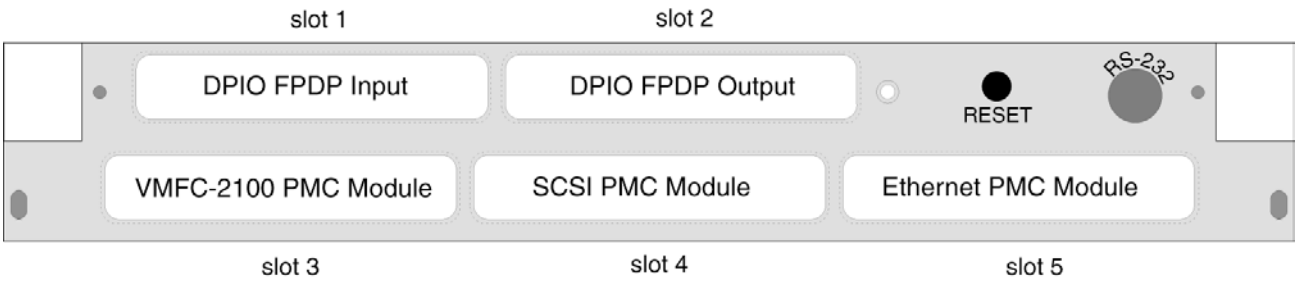
2.2 MDR Module Board Layout

The MDR module is shipped with the PMC Module(s) already mounted on the MDR-220 or MDR-250 board.





MDR-220 Top View & Front Panel (with typical PMC population)

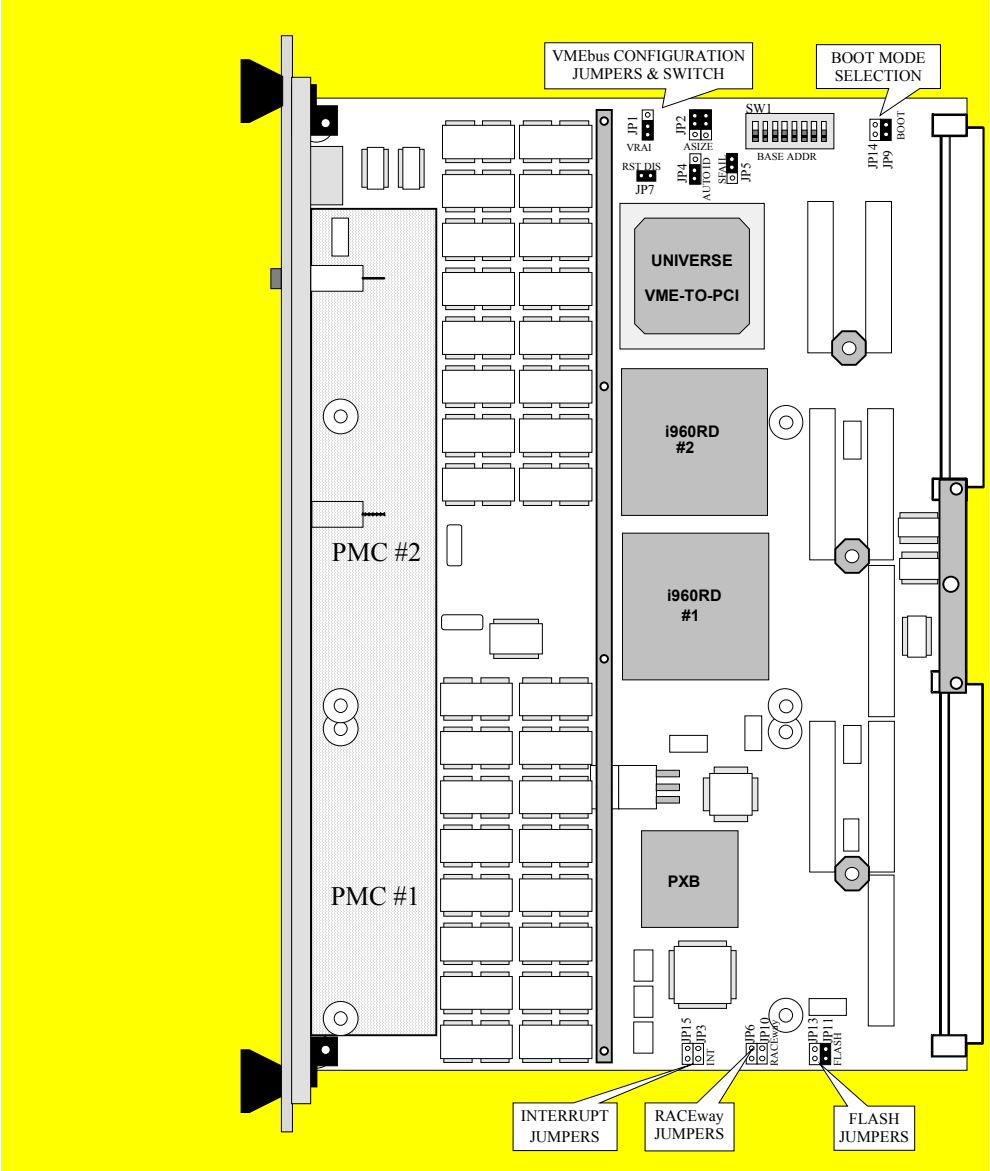


MDR-250 Front Panel (with typical PMC population)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.3 MDR Board Switch and Jumper Locations

The default switch and jumper settings on the MIDAS boards for the MDR base units at shipment are as shown in the figure below:



Note that darkened rectangles indicate the jumper is installed. For a complete discussion of each jumper and switch, refer to the *MIDAS 100/200 Series User's Manual.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.4 Installing the MDR Module into the VME Chassis

WARNING: Do not install the board in a powered system!
The MDR board can be installed into any **VMEbus slot** in a 6U VMEbus chassis, **as long as the daisy-chains for the bus grant and interrupt acknowledge signals are continuous from slot#1** to the slot in which the MDR board is installed.
Installation into slot#1 of a VMEbus system is automatically detected, as specified in the VME64 specification. System controller functions are also enabled consequently.

2.5 Cable Installation

Depending on the PMC population of the MDR, connect the cables as follows:

1. The RS-232 cable to the MIDAS board. The RS-232 is mainly used for debug output during MDR operations.
2. The Fibre Channel cable(s) between the VMFC-2100 PMC module(s) (slot 3-4) and the storage device(s),
3. The FPDP cable(s) between the MDR DPIO-XI/O module(s) (slot 1-2) and your own FPDP module(s),
4. The Ethernet cable between the Ethernet PMC module (slot 5) and your own Ethernet cable,
5. The SCSI cable between the SCSI PMC module (slot 4) and your tape drive,

2.6 Power Consumption

WARNING: Due to its power consumption, the MDR hardware requires forced air cooling for reliable operation. Operation on extender boards is not recommended.

MDR-220: Typical 3.5A/17.5W
MDR-250: Typical 3.5A/17.5W

Please see the sections for the relevant PMCs for their respective power consumption.

2.7 Hardware Configuration

The MDR Base Units are shipped in a condition that is least likely to interfere with existing VME and/or Ethernet installations. It is configured to not participate in any external network. As such, the only way to interact with the MDR initially is through the console serial port. Once the MDR Server is verified to run properly in the standalone mode, the files in the MDR FLASH may be edited for other options and configurations. See Chapter 10.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Standalone MDR

The MDR system is delivered as a pre-programmed ready-to-run system. In this configuration the built-in RS-232 serial port and user interface enables Standalone operation from a VT100 terminal or terminal emulator.

3.1 MDR Software

The MDR is shipped with several software modules preloaded into FLASH. These modules consist of the firmware, `mdrvxst.hex`, and MDR configuration files. The MDR configuration files are the following:

- **`mdr.ini`** MDR-specific configuration parameters file
- **`vxbsp.ini`** VxWorks BSP configuration file
- **`vxbp.txt`** VxWorks boot parameters file
- **`mmon.ini`** MIDAS Monitor configuration file

See Chapter 10: *Configuration Files* for complete details and modification procedures.

3.2 VxWorks Boot Parameters for the MDR board

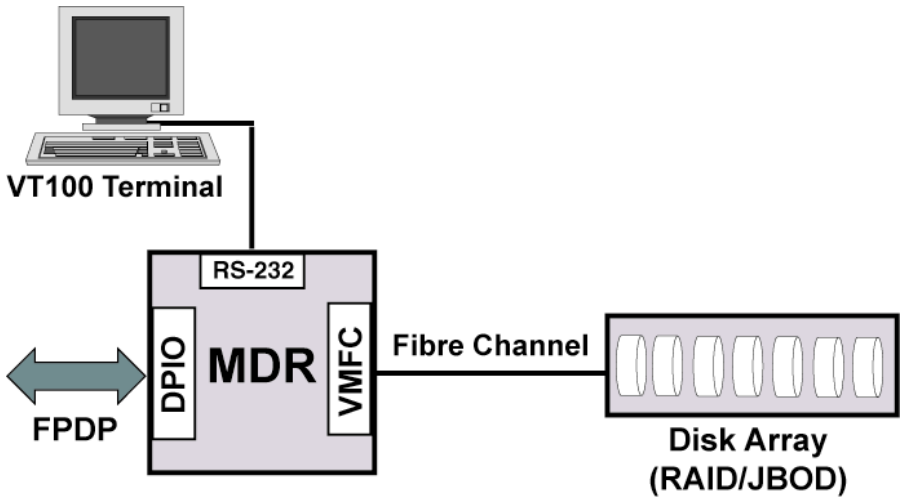
Both MDR boards, MDR-220 and MDR-250, run the VxWorks operating system. To boot VxWorks, several boot parameters must be provided. Each of the boot parameters is described below. More details may be obtained in the *Wind River Systems, Tornado Users Guide.*

`boot device`	Type of device to boot from. For standalone operation this must be set to a period. If a master controller is present, entry should be set to `mdcX`, where `X` is the slot number.
`Processor number`	A unique identifier for the target processor in a multiprocessor environment.
`host name`	Name of the host computer to boot from. This host is the location of the software image.
`file name`	Full path to the VxWorks object module stored within the boot host computer.
`inet on ethernet (e)`	Internet address of the target when an Ethernet interface is used.
`inet on backplane (b)`	Internet address of the target when a backplane interface is used.
`host inet (h)`	Internet address of boot host computer.
`gateway inet (g)`	If the host is not on the target network, this indicates address of the gateway node.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`user (u)`	User name that VxWorks uses to access host. User must have read access to boot-image file.
`ftp password (pw)`	User password. If omitted, then RSH is used instead of FTP.
`flags (f)`	Configuration options. Multiple options can be enabled using hexadecimal sum of values. 0x01 Do not enable system controller 0x02 Load all VxWorks symbols 0x04 Do not auto-boot 0x08 Auto-boot fast (3 count) 0x20 Disable login security 0x40 Use BOOTP to get boot parameters 0x80 Use TFTP to get boot image 0x100 Use proxy ARP
`target name (tn)`	name of target to be added to host table.
`startup script (s)`	path of startup script to execute after boot (available only if target-resident shell is included in download image).
`other (o)`	Normally not used.

3.3 Configuring the MDR for Standalone Operation



To configure any MDR unit to operate as a standalone system, allow the MDR unit to start up and then use the `bootChange` command to set the boot parameters as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
boot device             : sm
processor number        : 0
host name               : .
file name               : .
inet on ethernet (e)    : .
inet on backplane (b)   : <Backplane IP address>
host inet (h)           : .
gateway inet (g)        : .
user (u)                : .
ftp password (pw) (blank = use rsh): .
flags (f)               : .
target name (tn)        : <Target name>
startup script (s)      : .
other (o)               : .
```

For standalone operation, the `boot device` parameter must be set to a period "**.**"

`<Target name>` should be set to the MDR Server hostname that will be used to communicate with the MDR system.
`<Backplane IP address>` any available IP address can be used.

The default value for `flags` is 0. Do not change the value of `flags` – type a period "**.**" for the default value. For standalone operation, the `boot device` parameter must be set to a period "**.**"

3.4 Connecting to the MDR Server from the MDR Shell

Make sure that the RS-232 terminal or PC with a terminal emulator program is connected to the MDR.

To start the **`mdrshell`** program on the standalone MDR system:

1. Boot up the MDR and wait for the user prompt to appear:
   ```
   ->
   ```
2. At the user prompt, type:
   ```
   -> mdrshell
   ```
3. At the mdrshell prompt, type period "**.**":
   ```
   Specify the MDR Server name/IP address: .
   Trying to connect to server . …
   Connection with server . established.


   Got call from 192.168.15.101
   No current client session exist. Accept new client 13.
   ```
4. When the prompt reappears, press the Enter key:
   ```
   MDR Server hostname or IP address: ↵

   Welcome to VMETRO (RX-)MDR Shell Version 1.4.4
   Type <?> for help

   No current recording >
   ```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 Converting from Network to Standalone MDR

To convert the MDR from Host based to Standalone, the following procedure may be used. Essentially, the user downloads the file `vxbp-nonet.txt` from the Host in place of the vxbp.txt file. This action configures the board for standalone operation.

1) On the MDR board, enter the Midas Monitor and delete vxbp.txt:

```
# midel vxbp.txt
```

2) On the Host, go to `<path>/mdr/etc`, and retrieve the size of the file `vxbp-nonet.txt` (in bytes).

3) On the MDR board, type the following at the Midas Monitor prompt:

```
# mgf vxbp.txt <size>
```

Note: Use `vxbp.txt` and not `vxbp-nonet.txt`. This is because vxbp.txt is the name of the target file on the board, while `vxbp-nonet.txt` is the file on the Host to be copied.

4) Within HyperTerminal, click on Transfer->Send, set Protocol=Xmodem and Browse your computer to fetch file `<path>/mdr/etc/vxbp-nonet.txt`. Then click on Send. When the transfer is done, type 'quit' to reboot the board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

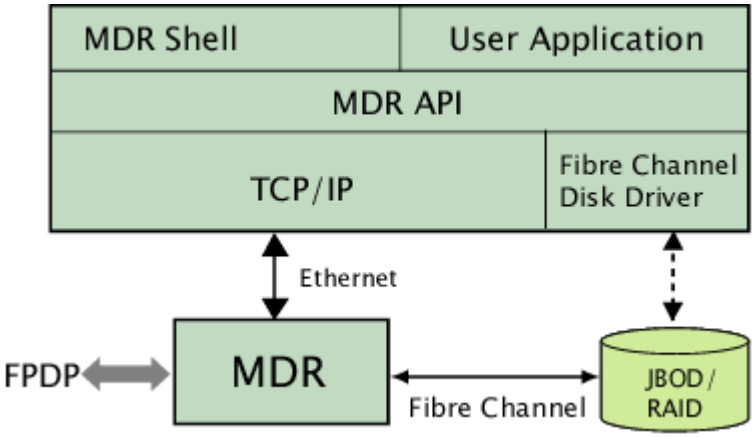
4 Host Network-Based MDR

In Host Network-based MDR, the Host computer communicates with the MDR using TCP/IP. The Host runs the MDR Shell (client) and the MDR board runs the MDR Server. The connection is TCP/IP over either Ethernet or Shared-Memory Network. The Ethernet option occupies a PMC slot; therefore, the MDR-250 is required. In the Shared-Memory Network option, a separate VME SBC is needed to provide a network gateway.

4.1 MDR Host Software

The two main components of the MDR host software package are the MDR Shell and the MDR API (Application Programming Interface). The MDR Shell is one of the manual user interfaces of the MDR. It provides a "command line" interface to the MDR. By normal MDR operations, we mean functions such as setting parameters of a recording, starting a recording, showing the state of the recorder and copying recordings.

The MDR API is an Application Programming Interface for application programs that need to control the MDR. The MDR API consists of essentially the same functions as the MDR shell and communicates with the MDR server in a client-server fashion.



Software structure on MDR host

Currently VxWorks 5.3.1/5.4 (PowerPC processor), Solaris 2.6/7/8 and Windows NT/2000 are supported.

The MDR unit may be operated with or without an external network. Before accessing the MDR unit from a networked host, some network configuration parameters have to be set up. This section describes how to set up the user-definable MDR parameters to support an external network.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To operate the MDR via the remote shell application (`mdrshell` on Solaris and `mdrshell.exe` on Windows), an external network (typically Ethernet) is required. On MDR-250 units, the interface may be provided by an Ethernet module purchased as one of the PMC options installed on the MDR unit itself.

With MDR-220 units, the interface to the external network must be provided by an additional SBC using Wind River Systems' Shared-Memory Network (for instance, an SBC running VxWorks 5.4), configured as the SM Network master. A MIDAS-120 with Ethernet PMC may be used in this capacity.

4.2 Host MDR Shell and API Installation

To install the Host MDR Shell program, insert the CD-ROM containing the software into the CD-ROM drive. If the software installation program does not start up automatically, run the **`Setup.exe`** program file. Follow the on-screen instructions to install the MDR software. For further details on installing VMETRO software, see the *VMETRO Products Installation Guide* shipped together with the CD-ROM.

All necessary files will be copied to a specified directory. Inside this directory, the structure is as follows:

`documents/`	contains the MDR documentation and example programs
`mdr-server/bin/`	contains the MDR Server binary file for all the supported platforms.
`mdr-server/etc/`	contains the configuration files for MDR Server `mdr.ini`, `mmon.ini`, `vxbsp.ini`, `vxbsp-min.txt`, `vxbsp-netif.txt` and `vxbsp-nonet.txt`.
`mdr-host-sw/bin/`	contains the MDR executable files for all the platforms specified above.
`mdr-host-sw/doc/`	contains the example programs
`mdr-host-sw/etc/`	contains the configuration file for MDR Host Software `mdr.ini`
`mdr-host-sw/include/`	contains the header files
`mdr-host-sw/lib/`	contains the MDR API libraries for all platforms described above.
`mdr-host-sw/fibre-channel-drivers/`	contains the VMFC-2X00(P) QLogic drivers for Windows NT/2000 and Solaris 2.6/7/8.

For those users who have purchased a VMFC-2X00 Fibre Channel PMC (or a PCI card) to be used for accessing the MDR FC disk subsystem directly from the host (Windows and/or Solaris), please refer to *Appendix B: Installing the Fibre Channel driver for Windows or Solaris*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.3 Configuring the MDR Shell

As part of the configuration, the MDR host software needs to read some parameters from a configuration file called `mdr.ini`. This file is stored in the `mdr-host-sw/etc` directory. In order for the MDR host software to know where to find this file, the environment variable `MDR_BASE` needs to point to the base of the MDR software installation. Therefore, the user has to declare an environment variable called `MDR_BASE`. This variable specifies the base directory of the 'MDR tree'. The `mdr.ini` file is expected to be in the sub-directory called `etc/` of the directory specified by `MDR_BASE`.

In the UNIX csh environment, `MDR_BASE` may be set with the command:

```
#setenv MDR_BASE /<mdr directory path>/mdr
```

In the Windows environment, `MDR_BASE` may be set with the command:

```
C:\>set MDR_BASE=/<mdr directory path>/mdr
```

In the Power PC (VxWorks) environment, MDR_BASE may be set with the command:

```
->putenv ("MDR_BASE=/<mdr directory path>/mdr")
```

In the UNIX environment, the path to the file `libmdrapi.so.0` (present in `mdr-host-sw/lib/sun-solaris/.libs`) must be added to the `LD_LIBRARY_PATH` environment variable.

In the Windows environment, the path to the files `mdrapi.dll` and `cygwin32.dll` (present in `mdr-host-sw/lib/x86-win32`) must be added to the `PATH` environment variable.

The MDR is shipped without any server lines defined in the `mdr.ini` file because the layout of the network is unknown. The user may optionally define servers to which the MDR Shell will attempt to connect automatically. The same is true for the `mdr.ini` file installed on the MDR unit itself. Without any server lines, the resident MDR shell will prompt for the "MDR Server hostname or IP address:" (i.e. the name or IP address of the MDR unit).
To define a server line for the MDR resident shell, reset the MDR, enter the MIDAS Monitor and type the command:

```
#iset Servers sX <IP addr> mdr.ini
```

where `X` is a number starting at 1 and `<IP Addr>` is the IP address for the MDR unit.

The `mdr.ini` file looks something like this:

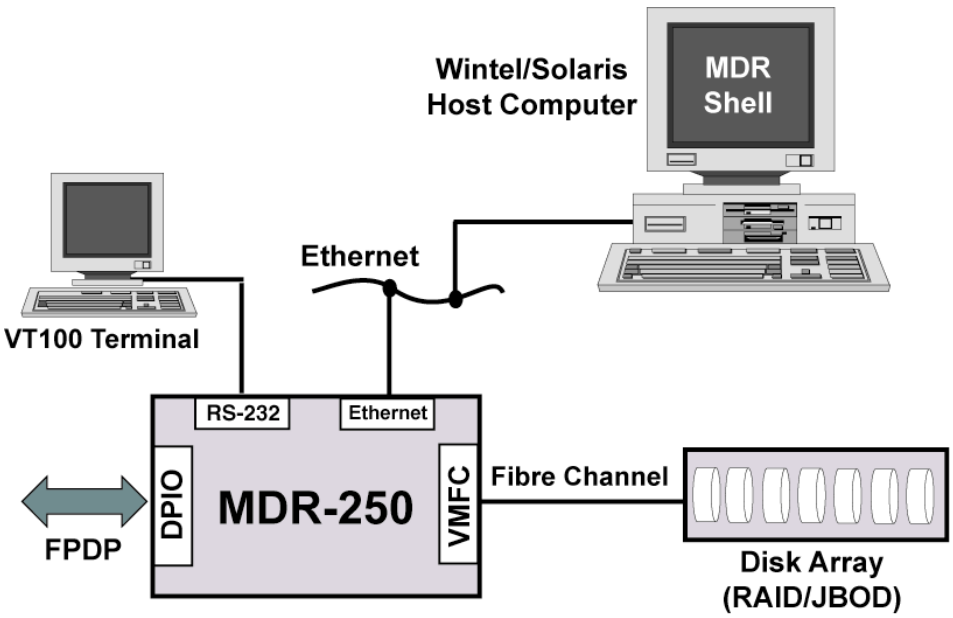
```
# MDR configuration file

[Servers]
## s1,s2,...sn specifies which MDR servers to communicate with
s1=midas
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The values of items `s1,s2,…sn` may specify the hostname(s) of one or several MDR systems that the user want to control with the MDR shell program. In the above example, only one MDR called `midas` is specified.
All hostnames/IP addresses in the `[Servers]` section have to be located in the hostname database being used by the computer running the MDR shell. The MDR shell program retrieves the IP address corresponding to the hostname for each host.

If the user specifies one or more server items in the `[Server]` section, then the MDR shell will not prompt the user for the hostname of the MDR server before trying to connect to the MDR server.

4.4 Configuring the MDR for Ethernet Operation



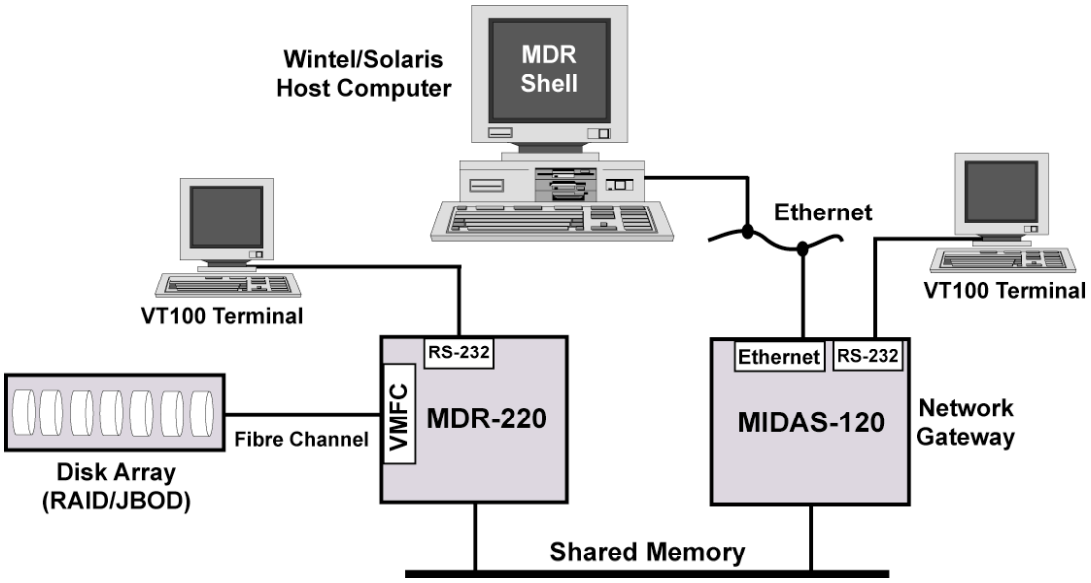
To configure an MDR-250-xxxxE unit with Ethernet operation, start up the MDR system and then use the `bootChange` command to set the boot parameters as follows:

```
-> bootChange

'.' = clear field; '-' = go to previous field; ^D=quit

boot device             : mdc5
processor number        : 0
host name               : .
file name               : .
inet on ethernet (e)    : <Ethernet IP address>
inet on backplane (b)   : .
host inet (h)           : .
gateway inet (g)        : .
user (u)                : .
ftp password (pw) (blank = use rsh): .
flags (f)               : 0x0
target name (tn)        : <Target name>
startup script (s)      : .
other (o)               : .
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

<slot> set to the slot location of the installed device
<Target name> should be set to the MDR Server hostname that will be used to communicate with the MDR system.
<Ethernet IP address> use an available IP address.

4.5 Configuring the MDR for Shared Memory Network Operation



To configure an MDR-220 to use an external network gateway provided by a MIDAS-120 equipped with an Ethernet PMC for example, allow the MDR system to start up and then use the bootChange command to set the boot parameters as follows:

```
boot device             : sm=0x10001100
processor number        : 1
host name               : .
file name               : .
inet on ethernet (e)    : .
inet on backplane (b)   : 192.168.0.101
host inet (h)           : .
gateway inet (g)        : 192.168.0.100
user (u)                : .
ftp password (pw) (blank = use rsh):
flags (f)               : .
target name (tn)        : <Target name>
startup script (s)      : .
other (o)               : .
```

The value sm=0x10001100 for boot device is just an example.
<procnum> processor number which is only used in Shared Memory network, should be set to a unique integer value for the VME system which is greater than 0 (the processor number 0 is reserved for the SM gateway).
<Target name> should be set to the MDR Server hostname that will be used to communicate with the MDR system.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The above Backplane IP address and Gateway IP address are valid examples – you may use IP addresses that are appropriate to your installation.

To configure the MIDAS-120 network gateway, allow the MIDAS system to start up and then use the `bootChange` command to set the boot parameters as follows:

```
boot device             : mdc<slot>
processor number        : 0
host name               : .
file name               : .
inet on ethernet (e)    : <Ethernet IP address>
inet on backplane (b)   : 192.168.0.100
host inet (h)           : .
gateway inet (g)        : .
user (u)                : .
ftp password (pw) (blank = use rsh):
flags (f)               : .
target name (tn)        : <Target name>
startup script (s)      : .
other (o)               : .
```

`<slot>` set to the slot location of the installed device
`<Target name>` should be set to the MDR Server hostname that will be used to communicate with the MDR system.
`<Ethernet IP address>` the IP address of the MIDAS-120 network gateway as seen from the Host Computer.

Verify that the `VmeA32SlaveBase` item in the `vxbsp.ini` file is set correctly. If this item is missing, then the VME A32 slave base address will be computed by the BSP.
The default BSP computation puts the VME A32 slave base of a MIDAS acting as a SM master (with a processor number of 0) at address 0x0.
For processor numbers different from 0, the following formula is used:

```
VmeA32SlaveBase = (ProcNum * 0x10000) + 0x4000000
```

In the above example configurations, the `VmeA32SlaveBase` for the MDR-220 should be set to 0x5000000 and that for the MIDAS-120 gateway should be set to 0x4000000.

The default value for `flags` is 0. Do not change the value of `flags` – type a period "**.**" for the default value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.6 Connecting to the MDR Server from the MDR Shell

To connect to an MDR board, the host computer and the MDR board must be on the same network. Also, the MDR board must be assigned a valid IP address. Also, ensure that `UseHostFcAdapter = 0` in the host `mdr.ini` file.

The `mdrshell` program can run on both a standalone MDR system and on a host computer. The difference between the two options is mainly the sets of commands available.

To start the **`mdrshell`** program on a host computer:

1. Locate the platform-type binaries directory:

   `Mdr-host-sw/bin/<type-of-platform>`

   where `<type-of-platform>` is one of the following: `midas-vxworks, sun-solaris, ppc-vxworks or x86-win32`.

2. Depending on your host platform, type:

   `mdrshell` (on a Solaris or VxWorks operating system)

   or run the

   `mdrshell.exe` file (on a Windows NT/2000 operating system).

If the hostname/IP address is not specified in the `mdr.ini` file, the `mdrshell` program will prompt the user for the hostname/IP address of the MDR Servers (until the user enters an empty hostname/IP address). This hostname needs to be a part of the host database on your system in order for the `mdrshell` program to resolve the IP address of the MDR Server. If you do not know how to determine the hostname/IP address of the MDR Server, ask your system administrator.
Make sure that the MDR Server is running and is connected to the network before you input the hostname of the MDR Server to the MDR Shell. If not, the MDR shell will display an error message and exit.
If the MDR Shell is successful in connecting to the MDR Server, the current state of the MDR server will be shown in the MDR Shell prompt, e.g.:

Example of MDR prompts:

```
Recording_1 (4.0 GiB) >
```

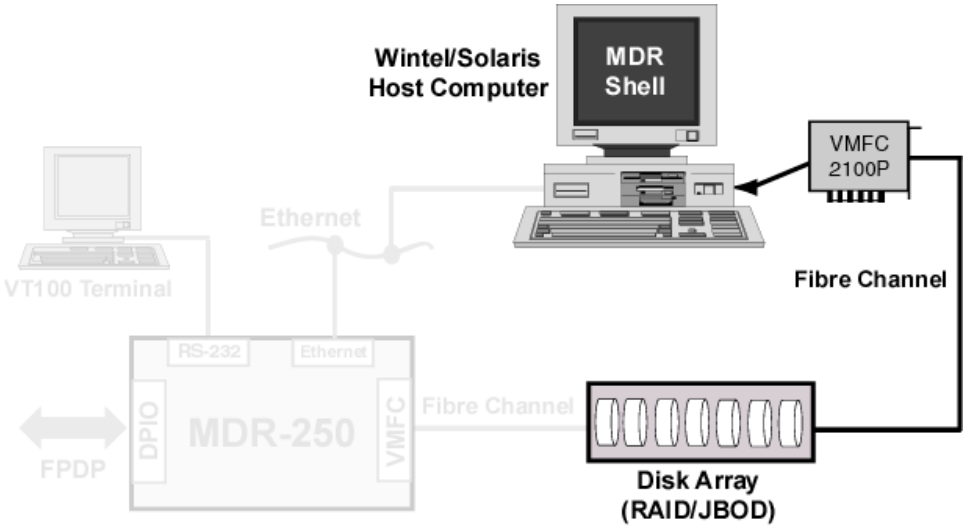
The syntax of the prompt is:

```
[/<current disk group>/]<current recording name> (<current recording size>) >
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5 Host Direct-Connection MDR

5.1 Configuring the MDR Host Software for Direct Connection



MDR provides several ways to read recorded data back from the disk storage system to a host system (or vice versa for preload in a playback application). It is possible to use Ethernet, Shared Memory Network or Fibre Channel.
For the fastest possible readback/preload of the data from/to the disk storage system, the storage device (RAID or JBOD) may be attached directly to the host, as indicated by the black line in the figure above.

This mode is available ONLY for Solaris and Windows NT/2000 Hosts (not available for VxWorks Hosts). Also, Direct Connection requires that the host is equipped with a Fibre Channel adapter (VMFC-2x00x-xx).

For more details on how to install the VMFC-2x00x-xx driver, please refer to the next section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2 Installing the Fibre Channel driver for Windows or Solaris

The Fibre Channel driver is optional. The drivers are located in the `mdr/mdr-host-sw/fibre-channel-drivers` directory on the VMETRO Software CD-ROM.

The MDR software has been tested with the VMETRO VMFC-2100(P) and VMFC-2200(P) boards for the following operating systems:

- Windows NT and Windows 2000
- Solaris 2.6, 7 and 8

However, the MDR Host Software relies on the QLogic driver (which uses the operating system and standard SCSI driver). Therefore, other FC boards using a standard SCSI FC driver may also work (although they are not supported by VMETRO).

5.2.1 Reprogramming the VMFC-2X00(P) board

In order to access directly the MDR Disk Subsystem, the MDR Host Software uses the Fibre Channel QLogic driver (for both Windows and Solaris). This driver requires the VMFC-2X00(P) board to have the QLogic identifiers for the System and Sub-System ids. Therefore, it is required to reprogram the EEPROM on the VMFC-2X00(P) using a QLogic tool.

Please follow the procedure below to reprogram your VMFC-2X00(P) board:

- On a Windows PC, install the VMFC-2X00(P) you want to reprogram
- Got to the directory called `mdr/mdr-host-sw/fibre-channel-drivers/vmfc-update`.
- Insert a blank floppy disk in the floppy disk drive and execute the program called `RawWrite`. In the Write tab, select the image file called `vmfc-update.img` and click on the Write button. This will copy the image file on the floppy disk. Wait for the operation to complete and exit the program.
- Reboot your computer (The floppy disk MUST still be located in the floppy disk drive) and boot on the floppy disk.
- Execute ql2100 if you have a VMFC-2100(P) board or ql2200 if you have a VMFC-2200(P). The QLogic tool should find you VMFC board (Typically, you should see "QLA2X00 adapter has been found …"). Choose the *Write NOVRAM* option (L option) in the menu. Wait for the completion of the following operations and check for any error message.
- If you don't see any error message then your VMFC-2X00(P) board has been updated successfully. Remove the floppy disk from its drive, shutdown the computer and remove the VMFC-2X00(P) board (if required).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.2 Installing the QLogic driver for Windows NT/2000

5.2.2.1 Install VMFC2X00 module

Install the VMFC2X00(P) module on your Windows NT/2000 computer or SBC. Connect the FC cable between the VMFC board and your disk unit.

5.2.2.2 Remove previously installed Fibre Channel driver

Before installing the new QLogic driver, it is essential to remove any installed FC driver (Fibre Gear driver from Delphi or older version of the QLogic driver).

5.2.2.3 Install the QLogic driver

If the MDR project is not already installed, follow the instructions in section 4.2: *MDR Shell and API Installation* to do so.

Go to the directory called
`<path>/mdr-host-sw/fibre-channel-drivers/x86-win32`

First, uncompress the file `qla2100_<OS>_<VersionNumber>.zip` (if you have a VMFC2100 board) or the file `qla2200_<OS>_<VersionNumber>.zip` (if you have a VMFC2200 board) on your Windows NT/2000 workstation.
Then follow the installation instructions provided by the `readme.pdf` file.

When the system comes back, make sure that the driver is running. To verify it, perform the following actions:

- Windows NT: go to the Control Panel, SCSI Adapters, Drivers tab. Check that the driver corresponding to your board (*QLogic QLA2000/2100 PCI Fibre Channel* or *QLogic QLA2200 PCI Fibre Channel*) is marked as *started*.
- Windows 2000: go to the Control Panel, System, Hardware tab, Device Manager, SCSI & RAID controllers. Check that the driver corresponding to your board (*QLogic 2100 PCI Fibre Channel* or *QLogic QLA2200 PCI Fibre Channel*) is indicated as *working properly*.

It is also recommended to check that Windows can locate the FC disks installed on your system. To verify it, perform the following actions:

- Windows NT: go to the Control Panel, SCSI Adapters, Devices tab. Check that the FC disks are present under the adapter.
- Windows 2000: go to the Control Panel, System, Hardware tab, Device Manager, Disk drives. Check that the FC disks are present. If the disks are not present then try the command *Scan for hardware changes* to update the list of disks.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.3 Installing the Fibre Channel QLogic Driver for Solaris

Install the VMFC2X00 module on your Solaris SPARC station or SBC.

5.2.3.1 Install the qla2X00 package

If the MDR project is not already installed, follow the instructions in section 4.2: *MDR Shell and API Installation*.

Go to the directory called `mdr-host-sw/fibre-channel-drivers/sun-solaris`

First you need to uncompress the file `qla2100_<VersionNumber>.zip` (if you have a VMFC2100 board) or the image file `qla2200<VersionNumber>.zip` (if you have a VMFC2200 board) on your Solaris SPARC station (or SBC) using the `unzip` tool.
Then you need to follow the instructions provided in the `readme.pdf` file.

As a summary:

- Uncompress the image file `qla2100.Z` (if you have a VMFC2100 board) or the image file `qla2200.Z` (if you have a VMFC2200 board) on your Solaris SPARC station (or SBC) using the `uncompress` tool.
- Become super-user.
- Install the qla2x00 driver package with the command line:

```
pkgadd -d <path>/qla2100
```

or

```
pkgadd -d <path>/qla2200
```

and follow the prompts. The default answers for the questions the installation program asks can be accepted. If the installation program asks to perform some actions, accept them.

- In the `/etc/driver_aliases` file, add the line `qla2100 "pci129a,2100"` if you have a VMFC-2100(P) board or `qla2200 "pci129a,2200"` if you have a VMFC-2200(P) board.

5.2.3.2 Prepare MDR Disk(s) for Solaris

- Connect fibre-channel disk(s) to the VMFC-2X00 host card,
- Use the `format` command and choose the appropriate disk, typically "`/dev/rdsk/c?t?d?`",
- Label the disk (`'l'` command),
- You may want to set the volume name (`'v'` command), e.g. '"`MDR-Vol1`"'.
  Note: This step is optional.

5.2.3.3 Running the MDR Shell

- In order to access the MDR storage device via FC connection, the MDR Shell program requires the administrator (or root) privileges. This is a requirement from the Windows and Solaris operating systems, not the MDR software.
- Set the `MDR_BASE` environment variable as shown in section 4.3: *Configuring the MDR Shell.*
- Run the MDR Shell program and just press the Enter key when prompted for the MDR server name.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.3 Connecting the MDR Shell to the disk subsystem

For more details on how to configure the MDR Host Software for direct read-back/preload, please refer to *Chapter 7: Configuration Files*.

Currently, the items to modify are present in the configuration file `mdr.ini`, section `StorageInfo` and are the following: `Disks`, `DiskTransferBlockSize`, `FcLoopRetryTime` and `UseHostFcAdapter`.

In order to establish a connection between the MDR Shell (running on the host software) and the MDR disk subsystem (as shown above), the user has to:

- Configure the items in the MDR host configuration file (`mdr.ini` file),
- Run the MDR Shell program (see section 4.3 for a more detailed description),
- Press `<Enter>` after the prompt: `MDR Server hostname or IP address`.

Remarks:

- In order to detect the FC disks, Windows and Solaris require the MDR Shell to have the administrator (or root) privileges.

In order to use the FC adapter installed in the Host computer, the `UseHostFcAdapter` parameter in the Host `mdr.ini` file must be set to '1'.

The MDR Shell then looks for the MDR disk subsystem and the prompt will appear, indicating that it is now connected to the MDR disk subsystem and ready to receive commands from the user.
If the configuration file items do not match the physical MDR disk subsystem configuration then an appropriate error message will be shown.

5.3.1 Unavailable MDR Shell commands

Because the MDR Shell is connected directly to the disk subsystem, the user does not have the possibility to use all the commands. The following commands are not available:

- `record, playback` and `stop`:
  A record/playback operation can be only be performed by the MDR Server.
- `selftest`:
  It is not currently possible to test the MDR disk subsystem from the MDR Shell in this configuration.
- `format-disk`:
  It is not currently possible to format the MDR disk subsystem from the MDR Shell in this configuration.
- `copy` (when using the parameters `disks tape` and `tape disks`):
  No tape device is attached to the host computer. Therefore, this operation is not possible.
- `reset`:
  This command applies only to the MDR Server. Therefore, this operation is not possible.

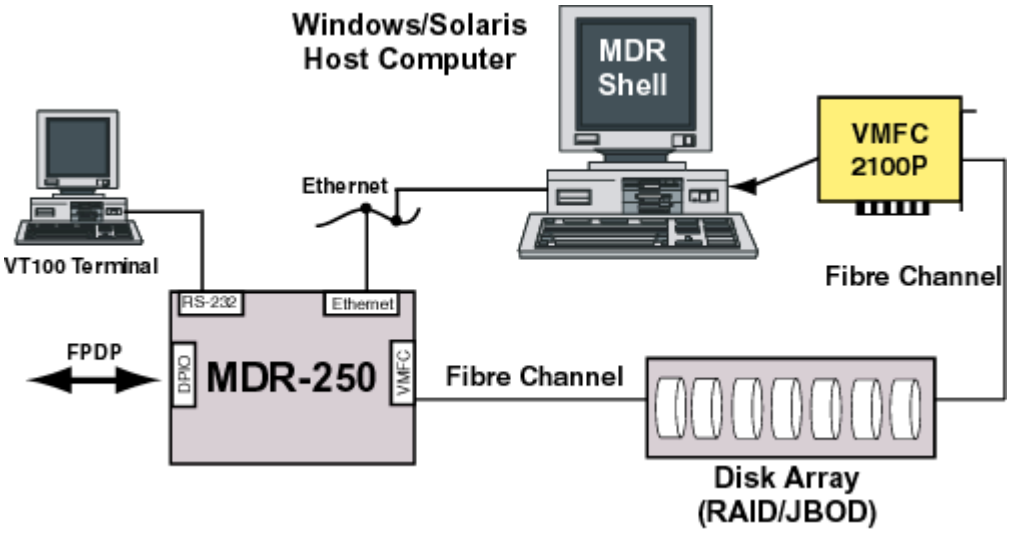
Note: The same restrictions apply to the MDR API functions corresponding to these MDR commands.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6 Host Network and Direct Connection MDR

Maximum functionality is available when both the Network and Direct Connection are present. The commands that are not supported or only partially implemented in the individual modes are fully functional in this combination mode.

6.1 Configuring the MDR for Network & Direct Connection



The configuration for this option is identical to Chapter 4: *Host Network-Based MDR*, except with the addition of the VMFC-2100P card as shown in Chapter 5: *Host Direct Connection MDR*. Addition of this card requires that the `UseHostFcAdapter` parameter in the Host `mdr.ini` file be set to '1'

Because of the lack of VxWorks support of Direct Connection, this mode is available ONLY for Solaris and Windows NT Hosts (not available for VxWorks Hosts).

It is possible to simultaneously record to and readback from a disk array using the Host Network & Direct Connection MDR mode. The MDR board records data to the disks while the Direct Connect reads back data. Recording speed is significantly impacted in this mode; therefore, the amount of data read back must be kept to a minimum. Note that tags must be enabled, because the readback operation must verify that the blocks read back are valid.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7 MDR Shell Commands

This chapter describes the commands available in the MDR Shell. Below is a list of the MDR Shell commands. In the last four 'configuration' columns, a 'Y' indicates fully supported, a 'P' indicates partial support, and a 'N' indicates not supported.

MDR Shell Commands	Description	Stand-alone	Host Network	Host Direct	Network & Direct
add-disks-to-disk-group	Add a list of disks to an existing disk group	N	N	Y	Y
clear-recording-table	Clear the MDR recording table	Y	Y	Y	Y
connect-disk-group	Connect to an existing disk group	P	P	Y	Y
copy-recording	Copy recorded data different devices	N	Y	Y	Y
create-disk-group	Create a disk group	N	N	Y	Y
delete-disk-group	Delete a disk group	N	N	Y	Y
delete-recording	Delete a recording	Y	Y	Y	Y
destroy-disk-group-info	Destroy the disk group information on a list of disks	N	N	Y	Y
diskspeedtest	Test recording speed of MDR system	N	Y	N	Y
dump-recording	Dump (display) recording data to screen	Y	Y	Y	Y
exit	Exit the MDR Shell program	Y	Y	Y	Y
format-disk	Low-level format the MDR disk subsystem	Y	Y	N	Y
help or ?	Display help information for MDR shell commands	Y	Y	Y	Y
list-fc-devices	List the fibre channel (FC) devices	P	P	Y	Y
new-recording	Allocate disk space for a new recording	Y	Y	Y	Y
playback	Start playback operation (MDR-250)	Y	Y	N	Y
prepare	Prepare a transfer (record or playback)	Y	Y	N	Y
record	Start record operation	Y	Y	N	Y
remove-disks-from-disk-group	Remove a list of disks from a disk group	N	N	Y	Y
rename-recording	Rename an existing recording	Y	Y	Y	Y
reset-MDR	Reset the MDR(s)	Y	Y	N	Y
resize-recording	Resize current recording	Y	Y	Y	Y
select-current-recording	Select recording as current	Y	Y	Y	Y
selftest	Test MDR disk subsystem data integrity	Y	Y	N	Y
set-start-block	Set start block of current recording	Y	Y	Y	Y
set-state	Set state of current recording	Y	Y	Y	Y
show-recording-table	Show the MDR recording table	Y	Y	Y	Y
stop	Stop the ongoing record operation	Y	Y	N	Y
tag-disable	Enable tags for a recording	Y	Y	Y	Y
tag-enable	Disable tags for a recording	Y	Y	Y	Y

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The `help` command will display a list of all MDR shell commands. Detailed information on each command may be displayed by typing:

```
help <CommandName>
```

Note that abbreviations are allowed as long as they are unique.
For instance: `res-r 50` for `resize-recording 50`

MDR Shell Command Description

add-disks-to-disk-group

Syntax:

```
add-disks-to-disk-group [<DiskGroupName>] <DiskIdList>
```

Description: Add a list of disks to an existing disk group.
The name of the disk group is specified by the optional argument `<DiskGroupName>`. If this argument is not specified then the current disk group name.
The disks to be added to the disk group are specified using a list (using commas ',' or dashes '-') containing their number (or identifier). The identifier of a disk can be retrieved using the `list-fc-devices` command: it corresponds to the number written in the first column.

Example:

```
add-disks-to-disk-group MyDiskGroup 1,2-4,7
add-disks-to-disk-group 4-8
```

clear-recording-table

Syntax:

```
clear-recording-table
```

Description: Clear the MDR recording table.
The `clear-recording-table` command will 'format' the recording table to an initial known state. Therefore, all the recordings will be lost.

Example:

```
clear
```

connect-disk-group

Syntax:

```
connect-disk-group [<DiskGroupId>]
```

Description: Connect the MDR Shell to an existing disk group. If the MDR Shell is connected to an MDR Server then this command connects also the MDR Server to the same disk group.
The identifier of the disk group to connect to is specified by the optional argument `<DiskGroupId>`. If this argument is not specified then the name of the disk group specified in the configuration file `mdr.ini` (Section `StorageInfo`, item `Disks`) is used.
In the special case that the MDR Shell is connected to an MDR Server but is NOT configured to access the MDR disk subsystem directly then it is possible to connect the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MDR Server to a new disk group using a list of disk port ids. For instance, `connect-disk-group 1,4-6` will connect the MDR Server to the disk group containing the disks with port ids 1,4,5 and 6.

Example:
```
connect-disk-group
connect-disk-group MyDiskGroup
```

copy-recording

Syntax:

```
copy-recording <Source> <Target> [<FileNamePathAndPrefix>]
[<RecordingName>] [<FirstRecordingBlock>-<LastRecordingBlock>]
```

Description: Copy the recorded data between different devices.
The source and destination devices, specified respectively by `<Source>` and `<Target>`, are one of the following:

`Disks`	MDR disk subsystem
`Host`	Host workstation (`ppc-vxworks, sun-solaris, x86-win32`)
`Tape`	Optional MDR SCSI tape drive (connected trough a SCSI PMC module)

The host file name path and prefix are specified by the `<FileNamePathAndPrefix>`. If not specified then the default value './' is used.
The name (or number in the recording table) of the recording to copy is specified by `<RecordingName>`. If this parameter is not specified, the current recording is copied.

Using the `copy-recording` command while making a recording affects the recording performance. The MDR Server may stop recording due to this.

Currently, only the following `copy-recording` commands are allowed:

```
copy-recording Disks Host [<FileNamePathAndPrefix>][<RecordingName>]
                  [<FirstRecordingBlock>-<LastRecordingBlock>]
```

Copy data from the MDR disk subsystem (recording specified by `<RecName>`) to the host.
The data will be written to one or more files in a locally mounted file system (platform x86-win32: the maximum size for a file is set to 1GiB). The directory of the file(s) may be specified with `<FileNamePathAndPrefix>`. In that case, `<FileNamePathAndPrefix>` must end with the subdirectory separator character for the host filesystem ('/' for UNIX, '\' for Windows NT).

The name of the file(s) will have the following format:
`<FileNamePathAndPrefix><RecordingName>.<ServerHostName>.<FileNo>`
where `<FileNo>` is the number of the file.
If `<FileNamePathAndPrefix>` is not specified then no prefix is added to the file name.
`<FirstRecordingBlock>` and `<LastRecordingBlock>` specify the range of recording blocks to copy (first and last user blocks included). If this range is not specified then the entire recording data is copied.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reading back data of a recording being recorded in infinite mode may end up in reading back data which will be (very) soon overwritten by the MDR.

`copy-recording Host Disks <FileName> [<RecordingName>]`

Copy data from host workstation file(s) (specified by `<FileName>`) to the MDR disk subsystem (recording specified by `<RecordingName>`). The `<RecordingName>` parameter must be the name (or number in the recording table) of an existing recording.

If the recording data is split into several files then the MDR software will try to copy as much data as possible. For instance on the `x86-win32` platform, if the recording size is 5 GiB and 5 files (named from `Recording1.MDR.0` to `Recording1.MDR.4`) are present on the host computer then the user can copy the 5GiB of data using the following command:

```
copy host disks Recording1.MDR.0 <RecordingName>
```

If the recording size is lower than the file size then only the recording size will be read from the file and copy to the recording. If greater, then the recording is truncated and the size is changed to the file size.

`copy-recording Disks Tape [<RecordingName>] [<FirstRecordingBlock>-<LastRecordingBlock>]`

Copy data from the MDR disk subsystem (recording specified by `<RecordingName>`) to SCSI tapes (one or several tapes according to the recording length and the internal drive compression rate).

`<FirstRecordingBlock>` and `<LastRecordingBlock>` specify the range of user blocks to copy (first and last user blocks included). If this range is not specified then the entire recording data is copied.

`copy-recording Tape Disks [<RecordingName>]`

Copy data from SCSI tapes (1 or several tapes according to the recording length) to the MDR disk subsystem (recording specified by `<RecordingName>`).

Examples:

```
copy-recording disks host ./data/today. Recording_1
copy-recording host disks ./today.Recording_1.mdr250.0 1
```

create-disk-group

Syntax:

```
create-disk-group <DiskGroupName> <DiskIdList>
```

Description: Create a disk group.

The name of the disk group is specified by the argument `<DiskGroupName>`.

The disks to be included into the disk group are specified using a list (using commas ‘,’ or dashes ‘-‘) containing their number (or identifier). The identifier of a disk can be retrieved using the `list-fc-devices` command: it corresponds to the number written in the first column.

Examples:

```
create-disk-group MyDiskgroup 1,2-4-7
Create-disk-group MyDiskgroup 4-8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

delete-disk-group

Syntax:

```
delete-disk-group <DiskGroupName>
```

Description: Delete a disk group.

The name of the disk group to delete is given by the `<DiskGroupName>` argument.

Examples:

```
delete-disk-group MyDiskGroup
```

delete-recording

Syntax:

```
delete-recording [<RecordingName>]
```

Description: Delete a recording.

The name (or number) of the recording to delete is given by the `<RecordingName>` argument. If the `<RecordingName>` is not specified then the current recording is deleted.

Examples:

```
delete-recording 1
delete-recording MyRecording
```

destroy-disk-group-info

Syntax:

```
Destroy-disk-group-info <DiskIdList>
```

Description: Destroy the disk group information on a list of disks.

The disks to be considered are specified using a list (using commas ',' or dashes '-') containing their number (or identifier). The identifier of a disk can be retrieved using the `list-fc-devices` command: it corresponds to the number written in the first column.

If this command is applied to a disk included in a group then the disk group information contained on this disk is **destroyed**. Therefore, the disk cannot be used in the disk group as is: it has to be re-added in the disk group using the `add-disks-to-disk-group` command.

Examples:

```
destroy-disk-group-info 1,2-4,7
```

diskspeedtest

Syntax:

```
diskspeedtest [<RecordingName>]
```

Description: Test the recording speed of the MDR system.

The name (or number) of the recording to test the MDR recording speed on is given by the `<RecordingName>` argument. If this argument is not specified then the current recording is used to perform the disk speed test.

Examples:

```
diskspeedtest 1
diskspeedtest MyRecording
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dump-recording

Syntax:

```
dump-recording [<BlockNo> [<RecName>]]
```

Description: Display recording data on the screen.

The `dump-recording` command reads a part of a recording and displays it to the screen.
`<BlockNo>` is the first recording block to be displayed. If the `<BlockNo>` parameter is not specified then the recoding block 0 is displayed.
`<RecName>` is the recording name (or number) of the recording to display. If this parameter is not specified then current recording data is displayed.

Different display options are offered during the command execution:

<+>, <n> or <Enter> will display the next page in the current block
<-> or <p> will display the previous page in the current block
<t> will toggle between little and big endian display mode
<o> will display from a specified offset (hexadecimal) in the block
<b> will display a specified block number (decimal)

Examples:

```
dump-recording 10
dump-recoding 0 Recording_1
```

exit

Syntax:

```
exit
```

Description: Exit the MDR Shell program.

A confirmation from the user is required.

Example: `exit`

format-disk

Syntax:

```
format-disk <DiskIdList>
```

Description: Low-level format for the MDR disk subsystem.

This command performs a low-level format of one or several disks (This operation may help to correct bad sectors on disks).
The disks to be formatted are specified using a list (using commas ‘,’ or dashes ‘-‘) containing their number (or identifier). The identifier of a disk should be retrieved using the `list-fc-devices` command: it corresponds to the number written in the first column.
When formatting several disks, the MDR system is able to format all the disks in parallel. Therefore, formatting one or several disks will take the same amount of time to complete. However, the `format-disk` command may take a long time to complete.

The `format-disk` command is a destructive command: all data stored on the MDR disk subsystem will be lost.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples:
```
format-disk 4
format-disk 4,6
format-disk 4-7
format-disk 4-11
format-disk 2,6-8,10
format-disk all
```

help or h or ?

Syntax:

```
help [<Command>]
```

Description: Display help information for MDR Shell commands.
If <Command> is not specified, then a list of all the commands is displayed.
If <Command> is specified, then a help text for the <Command> is displayed.

Examples:
```
help
help new-recording
help new-rec
```

list-fc-devices

Syntax:

```
list-fc-devices
```

Description: List the Fibre Channel devices (disks and disk groups).
Important information relative to each device is indicated as well as the devices included in the default group (marked as 'Default') and the devices included in the current group (marked as 'Connected')".

Example: `list-fc-devices`

new-recording

Syntax:

```
new-recording [<Blocks> [<BlockSize>]] [<InsertTag>]
```

Description: Allocate disk space for a new MDR recording.
The arguments <Blocks> and <BlockSize> specify the size of the new recording. The size is given in <Blocks> (as a decimal number) of size <BlockSize>.
The possible values for <BlockSize> are K or k (for KiB), M or m (for MiB), G or g (for GiB). If <BlockSize> is omitted then the block size is set by default to one mega binary byte (1MiB).
The argument <InsertTag> specifies if tags should be inserted during record operation.
The possible values for <InsertTag> are TAG (Insert tags during record operation) and NOTAG (do not insert tags during record operation).
The MDR Server will suggest a new recording name by appending _n to the current recording name (where n is a number that will be incremented for each new command).
The start block of the new recording will be moved to the block following the current recording block.
Without any parameters, the size will be fixed at the first free amount of space found on the disk subsystem. If the <Blocks> parameter is present then the MDR Server tries to find a space big enough on the disk subsystem to contain the recording.
If there is not enough space, the new command fails and no changes are made.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples:

```
new-recording
new-recording 5
new-recording 10 g
new-recording 3 G TAG
```

playback

Syntax:

```
playback [<Wait>] [<Infinite>] [<StartOffset>-<EndOffset>]
```

Description: Start playback operation.

The <Wait> parameter specifies whether to wait for the playback to finish or not. A value of WAIT will make the playback command wait for the playback to finish, whilst a value of NOWAIT will make the playback command not wait and return the user prompt immediately. If <Wait> is not specified, the playback command will wait for the playback to finish.

The <Infinite> parameter specifies if the playback operation is infinite or not. A value of INF will make the playback command to be infinite, whilst a value of NONINF will make the playback command to be non-infinite. If <Infinite> is not specified, the playback command will be started in non-infinite mode.

The <StartOffset>-<EndOffset> parameter specifies the interval of data in the recording to be played back. This interval is specified in bytes (hexadecimal). If this parameter is not specified then the entire recording data is played back (0-<Recording size>).

Examples:

```
playback
playback WAIT
playback NOWAIT NONINF
playback <4-0x80004>
```

prepare

Syntax:

```
prepare <RecordingName> <TransferType> [<Wait>] [<Infinite>]
[<StartOffset>-<EndOffset>]
```

Description: Prepare a transfer (record or playback operation)

The <RecordingName> parameter specifies the name (or number) of the recording to prepare.

The <TransferType> parameter specifies the type of transfer to prepare. A value of RECORD will prepare the specified recording for a record operation, whilst a value of PLAYBACK will prepares the specified recording for a playback operation.

The <Wait> parameter specifies whether to wait for the transfer to finish or not. A value of WAIT will make the record/playback command wait for the recording/playback to finish, whilst a value of NOWAIT will make the record/playback command not wait and return the user prompt immediately. If <Wait> is not specified, the record/playback command will wait for the recording/playback to finish.

The <Infinite> parameter specifies if the record operation is infinite or not. A value of INF will make the record command to be infinite, whilst a value of NONINF will make the record command to be non-infinite. If <Infinite> is not specified, the record command will be started in non-infinite mode by default.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The <StartOffset>-<EndOffset> parameter can only be used while preparing a playback operation. It specifies the interval of data in the recording to be played back. This interval is specified in bytes (hexadecimal). If this parameter is not specified then the entire recording data is played back (0-<Recording size>).

Examples:
```
prepare Recording_1 RECORD
prepare 1 PLAYBACK WAIT
prepare Recording_5 PLAYBACK 0x8-0x100000
prepare Recording_1 RECORD WAIT NONINF
```

record

Syntax:

```
record [<Wait>] [<Infinite>]
```

Description: Start record operation.

The <Wait> parameter specifies whether to wait for the recording to finish or not. A value of WAIT will make the record command wait for the recording to finish, whilst a value of NOWAIT will make the record command not wait and return the user prompt immediately. If <Wait> is not specified, the record command will wait for the recording to finish.

The <Infinite> parameter specifies if the record operation is infinite or not. A value of INF will make the record command to be infinite, whilst a value of NONINF will make the record command to be non-infinite. If <Infinite> is not specified, the record command will be started in non-infinite mode.

Examples:
```
record
record WAIT
record NOWAIT NONINF
```

remove-disks-from-disk-group

Syntax:

```
remove-disks-from-disk-group [<DiskGroupName>] <DiskIdList>
```

Description: Remove a list of disks from a disk group.

The name of the disk group to remove the disks from is specified by the optional argument <DiskGroupName>. If this argument is not specified then the current disk group is used.

The disks to be removed from the disk group are specified using a list (using commas ',' or dashes '-') containing their number (or identifier). The identifier of a disk can be retrieved using the list-fc-devices command: it corresponds to the number written in the first column.

Examples:
```
remove-disks-from-disk-group MyDiskGroup 1,2-4,7
remove-disks-from-disk-group 4-8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

rename-recording

Syntax:

```
rename-recording [<OldRecordingName>] <NewRecordingName>
```

Description: Rename an existing recording.

The <OldRecordingName> parameter specifies the current name of the recording to be renamed. If this parameter is not specified then the current recording is renamed.

The <NewRecordingName> parameter specifies the new recording name.

Examples:

```
rename-recording Recording_1 Rec1
rename-recording Rec2
```

Note:

The parameter <NewRecordingName> must **NOT** begin with a digit (0-9) but with a letter.

reset

Syntax:

```
reset [<ResetType>]
```

Description: Reset the MDR(s)

The <ResetType> argument specifies how the MDR(s) should be reset.

Before setting the value to the ResetType argument, it is important to understand that, depending on the MDR system hardware configuration (jumper settings), the reset may affect the behavior of the other boards present on the same VME backplane.

The hardware setting to consider is the jumper called SYSRESET DISABLE (RST DIS or JP7 on the diagram shown in *Appendix D: Switch and Jumper Settings*).

It is possible to reset the MDR system using the reset button at the front panel of the MDR system. This reset is called a hardware reset and is usually used when the MDR system is unable to communicate with the MDR host software because of a defective state.

When trying to reset the MDR system using the front panel reset button, the action of this jumper will be the following:

- Jumper RST DIS **removed:** Pushing the reset button will perform both a **local** reset (i.e. reset the MDR system by resetting the i960 processor) and a **global** reset (i.e. also potentially reset all the boards present on the same VME backplane). In this case, the SYSRST* signal on the VMEbus is asserted, causing all the VME boards to reset (if these boards are setup to reset when SYSRST* is asserted).
- Jumper RST DIS **inserted**: Pushing the reset button will only perform a **local** reset (PCI bus reset). This means that only the MDR system will be reset and this reset will not be propagated to the other VME boards present on the same VME back-plane. The behavior of these boards will therefore not be changed.

Without considering the hardware setting (RST DIS jumper position), the <ResetType> parameter can have two different values:

- VME_RESET Both **local** and **global** reset. Both local PCI buses and VME bus are reset.
- I960_RESET Only **local** software reset. The MDR system processor (Intel i960) is restarted without neither resetting the local PCI buses nor the VME bus. When the MDR Server program restarts, it tries to re-initialize all the PMC boards installed on

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the MDR system. However it is possible, in some cases (after a crash of one PMC board for instance), that the MDR Server is not able to re-initialize all the PMC boards. In that case, the MDR Server will report an error to the MDR host software during the connection phase (the connection will not be established). It is then necessary to reset the MDR system using the front panel reset button (or from the VxWorks shell).

If the <ResetType> parameter is not (or incorrectly) specified then the default value 1 is chosen. Therefore, a local software reset is carried out.

Considering the two configuration parameters of the reset command (jumper and <ResetType> parameter of the reset command), there are six different reset cases:

- Jumper setting: **in** (default MDR configuration)
  - Pushing the reset button on the front panel will reset the MDR system but not the other boards present on the same VME backplane,
  - Executing the 'reset VME_RESET' command from the MDR Shell will have no effect at all on the MDR Server(s),
  - Executing the 'reset I960_RESET' command from the MDR Shell will reset the MDR system but not the other boards present on the same VME backplane,
- Jumper setting: **out**
  - Pushing the reset button on the front panel will reset the MDR system and the other boards present on the same VME backplane,
  - Executing the 'reset VME_RESET' command from the MDR Shell will have exactly the same effect as pushing the front panel reset button,
  - Executing the 'reset I960_RESET' command from the MDR Shell will reset the MDR system but not the other boards present on the same VME backplane.

Examples:

```
reset
reset I960_RESET
reset VME_RESET
```

resize-recording

Syntax:

```
resize-recording <Blocks> [<BlockSize>]
```

Description: Resize the current recording.

The arguments <Blocks> and <BlockSize> specify the new size of the recording. The size is given in <Blocks> (as a decimal number) of size <BlockSize>.

The possible values for <BlockSize> are K or k (for KiB), M or m (for MiB), G or g (for GiB). If <BlockSize> is omitted then the block size is set by default to one mega binary byte (1MiB).

Examples:

```
resize-recording 5
resize-recording 10 g
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

select-current-recording

Syntax:

```
select-current-recording <RecordingName>
```

Description: Set the recording as current.

The <RecordingName> parameter specifies the name (or number) of the recording to select as current.

Examples:

```
select-current-recording Recording_1
select-current-recording 2
```

selftest

Syntax:

```
selftest [<Device>[:<DeviceSpecificParameters>]]
```

Description: Test data integrity of an MDR device.

This command performs a data integrity test of an MDR device. The MDR device is specified using the <Device> argument. The possible values for this argument are Disks (test the MDR Disk Subsystem) and Tape (test the MDR Tape Subsystem).

The <DeviceSpecificParameters> argument specifies the parameters relative to a test for a specific device.

No specific parameters are required for the MDR Tape Subsystem test:

```
selftest Tape
```

While testing the MDR Disk Subsystem, this command can be used with or without arguments:

- Without any arguments, the MDR will tests only the first stripe of all the disks. This command is used in that case to test the disk accesses (not the entire surface of the disks),
- With the argument Disks only, the MDR will test the same area as described above,
- With the argument Disks:<StartBlock>-<LastBlock>, where <StartBlock> represents the block where the test begins and <LastBlock> the block where the test ends. For example, to test the blocks from 0 to 8192 use the following command:

```
selftest Disks:0-8192
```

- With the arguments /Safe, the MDR disk test is non-destructive. For instance, to test the first 8192 blocks use the following command:

```
selftest Disks:0-8192/Safe
```

- With the arguments /Unsafe, the MDR disk test is destructive. For instance, to test the first 8192 blocks use the following command:

```
selftest Disks:0-8192/Unsafe
```

The selected blocks on the MDR disk subsystem will be checked according to the following sequence:

1. Save the data blocks on the disk to be checked if Safe was selected,
2. Write a pattern (random counting pattern) on these blocks,
3. Read back these blocks,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. Compare the two sets of blocks,
5. Restore the initial data blocks if `Safe` was selected,
6. Select a new set of blocks and go back to step 1.

Notes:

- The <StartBlock> parameter must be lower than the <LastBlock> parameter,
- If the <LastBlock> parameter is omitted then the MDR tests only the <StartBlock> block on the disks.

set-state

Syntax:

```
set-state <State>
```

Description: Set the state of current recording.

- The new state is given by the <State> argument. This argument can be set to STORED (the current recording stays or becomes stored) or NOTSTORED (the current recording stays or becomes not stored).

Example:

```
set-state STORED
set-state NOTSTORED
```

set-start-block

Syntax:

```
set-start-block <BlockNo>
```

Description: Set the start block of current recording.

The start block is given by the <BlockNo> argument.

Example:

```
set-start-block 400
```

Use caution when executing the `set-start-block` command as the recording may be inadvertently deleted. To avoid this when using the `set-start-block` command, always execute the commands for recording in the following sequence:

```
new-recording 10
set-start-block <BlockNo>
record
```

show-recording-table

Syntax:

```
show-recording-table
```

Description: Show the MDR recording table.

The show command displays the recorder information of the MDR recording table.

The show command prints out the following information:

```
Recorder info:
   Total capacity          : 67.8 GiB (  17365 blocks)
   Capacity used           :  0.0   B (      0 blocks)
   Capacity available      : 67.8 GiB (  17365 blocks)
   Largest recording size  : 67.8 GiB (  17365 blocks)
   Recorder block size     :  4.0 MiB
   Number of recordings    :    0     (maximum : 895)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`Total capacity` is the total capacity available on the MDR disk subsystem,
`Capacity used` is the total capacity allocated (for recording data) on the MDR disk subsystem,
`Capacity available` is the total capacity available on the MDR disk subsystem,
`Largest recording size` is the largest recording size available on the MDR disk subsystem,
`Recorder block size` is the size of each recorder block,
`Number of recordings` is the total number of recordings present on the MDR disk subsystem (stored of not).

Example: `show-recording-table`

stop

Syntax :

`stop`

Description: Stop ongoing record/playback operation.

In order to use the `stop` command, a record/playback operation must be started using the option `NOWAIT`, specifying that the MDR Shell should not wait for the record/playback operation to finish.

Example: `stop`

The end of the recording data may contain garbage data. The user has the responsibility to determine which part of the recording data is valid or not.

tag-disable

Syntax :

`tag-disable [<RecordingName>]`

Description: Disable tags for a recording

The `<RecordingName>` parameter specifies the name (or number) of the recording to disable the tags for.

Examples:

```
tag-enable
tag-enable Recording_1
tag-enable 5
```

tag-enable

Syntax :

`tag-enable [<RecordingName>]`

Description: Enable tags for a recording

The `<RecordingName>` parameter specifies the name (or number) of the recording to enable the tags for.

Examples:

```
tag-enable
tag-enable Recording_1
tag-enable 5
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8 MDR API Functions

To communicate with the MDR Server, the user has the possibility to use either the MDR Shell or his own program linked with the MDR API library, the MDR Shell program including the MDR API library.
The API functions are implemented as C++ functions that can be used in C++ programs. Below is an alphabetic list of all the MDR API functions available:

Name	Description
AddDisksToDiskGroup	Add a list of disks to a disk group
AllocateMemory	Allocate memory buffer
ClearDiskInfo	Clear the disk group information on a list of disks
ClearRecordingTable	Clear the recording table
ConnectToDiskGroup	Connect to a disk group
ConnectedToDiskGroup	Check a disk group connection
ConnectToServer	Set up an MDR client/server connection
ConnectedToServer	Check an MDR client/server connection
CopyRecording	Copy a recording
CreateDiskGroup	Create a new disk group
CreateNewRecording	Create a new recording
DeleteDiskGroup	Delete a disk group
DeleteRecording	Delete a recording
DisconnectFromDiskGroup	Disconnect from a disk group
DisconnectFromServer	Disconnect from the MDR server(s)
DiskSpeedTest	Test the MDR recording speed
FormatDisk	Format a disk
FreeMemory	Free memory buffer
GetDiskGroupName	Get the name of the current disk group
GetRecordingInfo	Get information from a recording
GetRecordingsInfo	Get information from all recordings
GetRecordingTableInfoString	Get the recording table information (as a string)
GetRecTableInfo	Get the recording table information
GetServerName	Get the server name
GetServerStatus	Get the server status
PrepareTransfer	Prepare a transfer
ReadRecorderBlock	Read block(s) from the storage device
ReadRecordingBlock	Read block(s) of a recording
RemoveDisksFromDiskGroup	Remove a list of disks from a disk group
ResetServer	Reset the MDR server(s)
Scan	Scan for fibre channel devices (disks and disk groups)
SelfTest	Test the MDR server system
SetRecordingInfo	Set information to a recording
SetRecTableInfo	Set the recording table information
StartTransfer	Start a transfer
StopTransfer	Stop a transfer
WaitTransfer	Wait for the current transfer to finish
WriteRecorderBlock	Write block(s) to the storage device
WriteRecordingBlock	Write block(s) of a recording

8.1 Header Files

In order to add the API functions into a C++ program, some MDR header files, located in the `mdr-host-sw/include` directory, have to be included. Some dependencies already exist between these files. Therefore, it is not required to add all the MDR header files but only those that are directly needed by the user's program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8.2 Error Conditions

Some of the functions may fail during execution and return an ERROR or NULL value, depending on the return type. If a function fails, the errno variable may be inspected to retrieve the cause of the failure. Some of the functions use standard errno values found in the errno.h file. The MDR specific errno values are found in the MdrDefs.h file.
Windows programs may need to use the GetLastError function in order to retrieve the error value.

8.3 API Function Description

AddDisksToDiskGroup

Declaration:

```
STATUS AddDisksToDiskGroup (char      * DiskGroupName,
                            DISK_INFO * DiskInfoTbl,
                            UINT        DiskInfoTblSize,
                            BOOL        ClearRecordingTable);
```

Description: Add a list of disks to an existing disk group.
This function adds one or several disks to an existing disk group. The disks to be added are specified using their information (port and controller identifiers). The disk information can be retrieved using the Scan function.
When adding a list of disks to an existing disk group, the recording table contained on the disk group is cleared or kept intact according to the value or the parameter ClearRecordingTable. If the recording table is cleared then any recording data present on the initial disk group or on the disks to add will be **LOST**.
Returned value: OK or ERROR.
If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

AllocateMemory

Declaration:

```
STATUS AllocateMemory (void ** DataBuffer,
                       UINT    DataBufferSize);
```

Description: Allocate memory buffer.
This function allocates a memory buffer required by the ReadRecorderBlock, ReadRecordingBlock, WriteRecorderBlock and WriteRecordingBlock functions.
AllocateMemory tries to allocate <DataBufferSize> bytes of memory. If the memory allocation is successful then the base address of the allocated buffer is stored in the memory address pointed to by <Buffer>.
Returned value: OK or ERROR.
If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ClearDiskInfo

Declaration:

```
STATUS ClearDiskInfo (DISK_INFO * DiskInfoTbl,
                      UINT       DiskInfoTblSize);
```

Description: Clear the disk group information on a list of disks.

This function clears the disk group information saved on the specified disks. The disks used during this operation are specified using their information (port and controller identifiers). The disk information can be retrieved using the `Scan` function.

After completion of this function, these disks will **NOT** contain any disk group information. Therefore, any disk group information present on the disks will be **LOST**.

In a general way, this function should only be used when a disk has been extracted from a disk group and cannot be inserted into a new disk group (because it still contains the information of the previous disk group). Clearing the disk information on the disk fixes the problem.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

ClearRecordingTable

Declaration:

```
STATUS ClearRecordingTable (void);
```

Description: Clear the recording table.

This function clears the recording table by deleting all the recordings.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

ConnectToDiskGroup

Declaration:

```
STATUS ConnectToDiskGroup (char * DiskGroupId,
                           UINT   Options =
                               MDR_DISK_GROUP_CONNECT_CLIENT,
                           UINT   DiskStripeSize = 0);
```

Description: Connect to an MDR Disk Group.

`ConnectToDiskGroup` tries to establish a connection between the MDR Client (or MDR Server) and the MDR disk group specified by `DiskGroupId`.

The `Options` parameter specifies which MDR actor (Client and/or Server) should be connected to the disk group:

- `MDR_DISK_GROUP_CONNECT_CLIENT`: Only the MDR Client shall be connected to the disk group. This option will be refused by the MDR Host Software if the MDR Client is already connected to an MDR Server (the MDR Server shall also be connected to the same disk group)
- `MDR_DISK_GROUP_CONNECT_SERVER`: Only the MDR Server shall be connected to the disk group. This option will be refused by the MDR Host Software if the MDR Client is already connected to a disk group (the MDR Client shall also be connected to the same disk group)
- `MDR_DISK_GROUP_CONNECT_CLIENT || MDR_DISK_GROUP_CONNECT_SERVER`: Both MDR Client and Server shall be connected to the disk group.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The DiskStripeSize parameter specifies the stripe size to be used in the disk group. This parameter should only be used when connecting an MDR Server alone to a static disk group.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

ConnectedToDiskGroup

Declaration:

```
BOOL ConnectedToDiskGroup (char * DiskGroupName);
```

Description: Check the connection to the MDR Disk Group specified by the DiskGroupName parameter.

Returned value: TRUE if the client is connected to the disk group DiskGroupName and FALSE if it is not.

ConnectToServer

Declaration:

```
STATUS ConnectToServer (char * ServerHostname);
```

Description: Connect to the MDR server.

ConnectToServer tries to establish a connection between the MDR_CLIENT object and the MDR server specified by ServerHostname.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

ConnectedToServer

Declaration:

```
BOOL ConnectedToServer(void);
```

Description: Check the connection to the MDR server.

ConnectedToServer checks if the client is connected to an MDR server.

Returned value: TRUE if the client is connected to an MDR server and FALSE if not.

CopyRecording

Declaration:

```
STATUS CopyRecording (UINT  RecordingNb,
                      char *Source,
                      char *Target,
                      UINT  FirstRecordingBlockToCopy,
                      UINT  LastRecordingBlockToCopy);
```

Description: Copy a recording.

This function copies the data of a recording between source and target devices.

The source and target devices, specified respectively by <Source> and <Target> are one of the following:

Disks	MDR disk subsystem
Tape	SCSI tape drive (connected trough a SCSI PMC module)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The number of the recording to copy is specified by the parameter RecordingNb. If this parameter is not specified then the current recording is copied.
While copying data from the MDR Disks Subsystem to the MDR Tape Subsystem, it is possible to specify which part of the recording is copied using the FirstRecordingBlockToCopy and LastRecordingBlockToCopy parameters. These parameters represent the first and last blocks of the recording to be copied.
For instance, to copy all the data contained in a recording, set the parameter FirstRecordingBlockToCopy to 0 and the parameter LastRecordingBlockToCopy to the number of user blocks (field UserBlocks of the RECORDING structure) minus 1.

Using the CopyRecording function while making a recording affects the recording performance. The MDR Server may stop recording due to this.

Currently, only the following copy commands are allowed:

Source	Target	
Disks	Tape	copy data from the MDR disk subsystem to SCSI tapes (1 or several tapes according to the recording length and the internal drive compression rate),
Tape	Disks	copy data from SCSI tapes (1 or several tapes according to the recording length) to the MDR disk subsystem,

Returned value: OK or ERROR.
If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

CreateDiskGroup

Declaration:

```
STATUS CreateDiskgroup (char        DiskGroupName,
                        DISK_INFO * DiskInfoTbl,
                        UINT        DiskInfoTblSize,
                        UINT        DiskStripeSize,
                        BOOL        ClearRecordingTable);
```

Description: Create a new disk group.
This function creates a new disk group named <DiskGroupName>. The new disk group will contain the disks whose information is located in the <DiskInfoTbl> parameter. The disks are specified using their information (port and controller identifiers). Disk information can be retrieved using the Scan function.
The <DiskStripeSize> parameter defines the size of each stripe on each disk. Therefore, the size of a stripe on the disk group is the <DiskInfoTblSize> parameter (number of disks in the disk group) times the <DiskStripeSize> parameter.
Currently, 512KiB (512*1024 bytes) is the only value supported by the MDR Software for the <DiskStripeSize> parameter.
When adding a list of disks to an existing disk group, the recording table contained on the disk group is cleared or kept intact according to the value or the parameter ClearRecordingTable. If the recording table is cleared then any recording data present on the initial disk group or on the disks to add will be **LOST**.
Returned value: OK or ERROR.
If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CreateNewRecording

Declaration:

```
STATUS CreateNewRecording (RECORDING * Recording);
```

Description: Create a new recording.

This function creates a new recording (according to the `Recording` parameter) and makes it the current recording.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

DeleteDiskGroup

Declaration:

```
STATUS DeleteDiskGroup (char * DiskGroupName);
```

Description: Delete a disk group.

This function deletes an existing disk group from the system and removes all the disks included in the disk group (Clear the disk group information on the disks).

Therefore, any disk group information present on the disks composing the disk group will be **LOST**.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

DeleteRecording

Declaration:

```
STATUS DeleteRecording (UINT   RecordingNo);
```

and

```
STATUS DeleteRecording (char * RecordingName);
```

Description: Delete a recording.

This function deletes the recording specified. Either of the recording number or name can be used to specify the recording in the recording table.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

DisconnectFromDiskGroup

Declaration:

```
STATUS DisconnectFromDiskGroup (char * DiskGroupName,
                                UINT   Options =
                        MDR_DISK_GROUP_DISCONNECT_CLIENT);
```

Description: Disconnect from an MDR disk group.

`DisconnectFromDiskGroup` tries to disconnect the MDR Client (and/or MDR Server) from the MDR disk group specified by `DiskGroupName`.

The `Options` parameter specifies which MDR actor (Client and/or Server) should be disconnected from the disk group:

- `MDR_DISK_GROUP_DISCONNECT_CLIENT`: Only the MDR Client shall be disconnected from the disk group.
- `MDR_DISK_GROUP_DISCONNECT_SERVER`: Only the MDR Server shall be disconnected from the disk group.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- MDR_DISK_GROUP_DISCONNECT_CLIENT || MDR_DISK_GROUP_DISCONNECT_SERVER: Both MDR Client and Server shall be disconnected from the disk group.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

DisconnectFromServer

Declaration:

```
STATUS DisconnectFromServer (void);
```

Description: Disconnect from the MDR server.

This function shuts down the session with the MDR server and puts the MDR_CLIENT object back into the initial state.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

DiskSpeedTest

Syntax:

```
STATUS DiskSpeedTest (UINT     RecordingNo,
                      UINT   * TransferRate);
```

Description: Test the recording speed of the MDR system.

The number of the recording to test the MDR recording speed on is given by the <RecordingNo> argument.

<TransferRate> represents the recording rate (in bytes per second) reached by the MDR system.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

FormatDisk

Declaration:

```
STATUS FormatDisk (DISK_INFO * DiskInfoTbl,
                   UINT        DiskInfoTblSize);
```

Description: Format one or several disks

The list of disks to be formatted is specified using their information. Disk information can be retrieved using the Scan function.

The DiskInfoTbl parameter is an array of disk information representing the disks to format. The number of disks to format is specified using the DiskInfoTblSize parameter.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The `FormatDisk` function is destructive: all data stored on the disks to be formatted will be lost.

It is possible to format several disks in parallel. Therefore, formatting one or several disks will take the same amount of time to complete. In order to format several disks in parallel, call the `FormatDisk` function for each disk to format without waiting between each call. After the last call, you need to wait for the end of the formatting phase by pooling the MDR Server status. The format operation is completed when this status changes from `MDR_SERVER_FOMATTING_DISK` to `MDR_SERVER_OK`.

FreeMemory

Declaration:

```
STATUS FreeMemory (void ** DataBuffer);
```

Description: Free allocated memory buffer.

This function frees a memory buffer previously allocated using the `AllocateMemory` function.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

GetDiskGroupName

Declaration:

```
const char * GetDiskGroupName (void);
```

Description: Get the disk group name.

This function returns the name of the current disk group the MDR client is connected to.

Returned value: A string or `NULL`.

If `NULL` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

GetRecordingInfo

Declaration:

```
STATUS GetRecordingInfo (UINT        RecordingNo,
                         RECORDING * Recording);
```

and

```
STATUS GetRecordingInfo (char        RecordingName,
                         RECORDING * Recording);
```

Description: Get information from a recording.

This function copy the recording information corresponding to the recording identified by `RecordingNo` into the `RECORDING` structure pointed to by `Recording`. Either of the recording number or name can be used to find the recording in the recording table. If 0 is specified as recording number, then the current recording information is returned.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetRecordingsInfo

Declaration:

```
int GetRecordingsInfo (RECORDING * RecordingArray,
                       UINT        RecordingArraySize);
```

Description: Get information of some recordings.

This function copy the recording information corresponding to the number of recordings specified by `RecordingArraySize` into the `RECORDING` structure pointed to by `Recording`. `RecordingArraySize` gives the size, in number of recordings, of the `RECORDING` output structure `RecordingArray`.

Returned value: Number of recordings put into `RecordingArray` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

GetRecordingTableInfoString

Declaration:

```
STATUS GetRecordingTableInfoString (char * DataBuffer,
                                    UINT   DataBufferLength);
```

Description: Get the information string of the recording table.

This function creates an information string of the recording table.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

GetRecTableInfo

Declaration:

```
STATUS GetRecTableInfo (REC_TABLE_INFO * RecTableInfo);
```

Description: Get the recording table information.

This function returns the `REC_TABLE_INFO` structure of the MDR and copies it into the location pointed to by `RecTableInfo`.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

GetServerName

Declaration:

```
const char * GetServerName (void);
```

Description: Get the server name.

This function returns a pointer to character string containing the hostname of the MDR server.

Returned value: A string or `NULL`.

If `NULL` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetServerStatus

Declaration:

```
STATUS GetServerStatus(void);
```

and

```
STATUS GetServerStatus(UINT32 * BootStatus);
```

and

```
STATUS GetServerStatus(MDR_OPERATION_STATUS * OperationStatus);
```

Description: Get the server status.

The first function requests the status of the MDR server. This function should be called after a successful call to the function `ConnectToServer` to check that everything is correct on the MDR server side.

The second function requests the boot status of the MDR Server. Please refer to the appendix G for more information on the MDR Server boot status.

The third function requests the operation status of the MDR server. Polling the operation status of an MDR server can be used to retrieve the progress of an operation (record, playback, copy, etc.). The `MDR_OPERATION_STATUS` structure contains the following fields:

- `Operation`: Current server operation (same as server state, see definitions in `MdrDefs.h` file)
- `SourceId`: Identifier of the source device (see definitions of source devices in `MdrDefs.h` file)
- `TargetId`: Identifier of the target device (see definitions of target devices in `MdrDefs.h` file)
- `BlocksToTransfer`: Number of blocks to be transferred during the current running transfer operation (if the current server operation is a transfer)
- `BlocksTransferred`: Number of blocks transferred so far during the current running transfer operation (if the current server operation is a transfer)
- `BlockSize`: Size of each block to be transferred during the current running transfer operation (if the current server operation is a transfer)

Returned value: status of the MDR Server. This value can be among the following values:

Status	Meaning
MDR_SERVER_OK (or OK)	Idle state
MDR_SERVER_ERROR (or ERROR)	Error has occurred
MDR_SERVER_TRANSFERRING	Transferring data (record or playback)
MDR_SERVER_INITIALIZING	Initializing a copy (using tape drive)
MDR_SERVER_COPYING	Copying data (using tape drive)
MDR_SERVER_WAITING_NEXT_TAPE	Waiting for the next tape
MDR_SERVER_WAITING_RIGHT_TAPE	Waiting for the right tape
MDR_SERVER_FORMATTING_DISK	Formatting one or several disks
MDR_SERVER_NOT_CONNECTED_TO_SD	Not connected to any disk group

The `errno` variable is set accordingly to the status returned. See `MdrDefs.h` for the definition of the status and the MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PrepareTransfer

Declaration:

```
STATUS PrepareTransfer (MDR_TRANSFER    * MdrTransfer,
                        MDR_TRANSFER_ID * TransferId = NULL);
```

Description: Prepare a transfer.

This function prepares a transfer (record or playback). The transfer is defined using the `MdrTransfer` structure. This structure is composed of the following fields:

- `RecordingNo`: number of the recording to be used during the transfer
- `TransferType`: type of transfer to carry out (`MDR_TRANSFER_TYPE_RECORD` for a record operation or `MDR_TRANSFER_TYPE_PLAYBACK` for a playback operation)
- `Infinite`: transfer operation is infinite (`MDR_TRANSFER_INFINITE_INF`) or not infinite (`MDR_TRANSFER_INFINITE_NONINF`)
- `Response`: MDR Client program will wait for the transfer to complete (`MDR_TRANSFER_RESPONSE_WAIT`) or will not wait for the transfer to complete (`MDR_TRANSFER_RESPONSE_NOWAIT`)
- `StartOffsetLow`: low part of the 64-bit offset of the first byte to be transferred (only valid for playback operation)
- `StartOffsetHigh`: high part of the 64-bit offset of the first byte to be transferred (only valid for playback operation)
- `EndOffsetLow`: low part of the 64-bit offset of the last byte to be transferred (only valid for playback operation)
- `EndOffsetHigh`: high part of the 64-bit offset of the last byte to be transferred (only valid for playback operation)

Refer to the header files for more information about the `MDR_TRANSFER` structure.

A transfer identifier, pointed to by `TransferId`, is used to identify a transfer while starting it, stopping it, etc. This identifier is computed by the MDR API and is an output parameter of this function.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

ReadRecorderBlock

Declaration:

```
STATUS ReadRecorderBlock (UINT   FirstRecorderBlockToRead,
                          UINT   RecorderBlocksToRead,
                          void * DataBuffer,
                          UINT   DataBufferSize);
```

Description: Read recorder block(s) from the storage device.

This function reads one or more recorder blocks from the storage device into memory. If the recording data (located in the recorder blocks) contains tags then these tags are read by the MDR and are therefore present in the data buffer.

The `FirstRecorderBlockToRead` parameter specifies the first recorder block to be read.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Using the `ReadRecorderBlock` function while making a recording affects the recording performance. The MDR Server may stop recording due to this.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Reading back data of a recording being recorded in infinite mode may end up in reading back data that will be (very) soon overwritten by the MDR.

ReadRecordingBlock

Declaration:

```
STATUS ReadRecordingBlock (UINT   RecordingNo,
                           UINT   FirstRecordingBlockToRead,
                           UINT   RecordingBlocksToRead,
                           void * DataBuffer,
                           UINT   DataBufferSize);
```

Description: Read user block(s) of a recording.

This function reads one or more user blocks of the recording specified by `RecordingNo`. If the recording data contains tags then these tags are **not** read by the MDR and therefore are **not** present in the data buffer. The `FirstRecordingBlockToRead` and `RecordingBlocksToRead` parameters map to the user block parameters in the `RECORDING` structure corresponding to the specified recording. Hence, the `RecordingBlockSize` specifies the size of the blocks.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Using the `ReadRecordingBlock` function while making a recording affects the recording performance. The MDR Server may stop recording due to this.

Reading back data of a recording being recorded in infinite mode may end up in reading back data that will be (very) soon overwritten by the MDR.

RemoveDisksFromDiskGroup

Declaration:

```
STATUS RemoveDisksFromDiskGroup (char * DiskGroupName,
                                 UINT * DiskPortIdTbl,
                                 UINT   DiskPortIdTblSize,
                                 BOOL   ClearRecordingTable);
```

Description: Remove a list of disks from a disk group.

This function removes a list of disks (specified by their port identifier) from an existing disk group (specified by its name).

When removing a list of disks from an existing disk group, the recording table contained on the disk group is cleared or kept intact according to the value or the parameter `ClearRecordingTable`. If the recording table is cleared then any recording data present on the initial disk group or on the disks to add will be **lost**.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ResetServer

Declaration:

```
STATUS ResetServer (UINT ResetType);
```

Description: Reset the MDR Module(s).

This function requests the MDR module to reset.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

The <ResetType> argument specifies how the MDR Server(s) should be reset.

Before setting the value to the ResetType argument, it is important to understand that, depending on the MDR system hardware configuration (jumper settings), the reset may affect the behavior of the other boards present on the same VME backplane.

The hardware setting to consider is the jumper called SYSRESET DISABLE (RST DIS or J7 on the diagram shown in the *Appendix D: Switch and Jumper Settings*).

It is possible to reset the MDR system using the reset button present in front panel of the MDR system. This reset is called a hardware reset and is usually used when the MDR system is unable to communicate with the MDR host software because of a defective state.

When trying to reset the MDR system using the front panel reset button, the action of this jumper will be the following:

- Jumper RST DIS **removed:** Pushing the reset button will perform both a **local** reset (i.e. reset the MDR system by resetting the i960 processor) and a **global** reset (i.e. also potentially reset all the boards present on the same VME backplane). In this case, the SYSRST* signal on the VMEbus is asserted, causing all the VME boards to reset (if these boards are setup to reset when SYSRST* is asserted).
- Jumper RST DIS **inserted**: Pushing the reset button will only perform a **local** reset (PCI bus reset). This means that only the MDR system will be reset and this reset will not be propagated to the other VME boards present on the same VME back-plane. The behavior of these boards will therefore not be changed.

Without considering the hardware setting (RST DIS jumper position), the <ResetType> parameter can have two different values:

- MDR_RESET_VME Both **local** and **global** reset. Both local PCI buses and VME bus are reset.
- MDR_RESET_I960 Only **local** software reset. The MDR system processor (Intel i960) is restarted without neither resetting the local PCI buses nor the VME bus. When the MDR Server program restarts, it tries to re-initialize all the PMC boards installed on the MDR system. However it is possible, in some cases (after a crash of one PMC board for instance), that the MDR Server is not able to re-initialize all the PMC boards. In that case, the MDR Server will report an error to the MDR host software during the connection phase (the connection will not be established). It is then necessary to reset the MDR system using the front panel reset button (or from the VxWorks shell).

If the <ResetType> parameter is not (or incorrectly) specified then the default value 1 is chosen. Therefore a local software reset is carried out.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Considering the two configuration parameters of the `reset` command (jumper and <`ResetType`> parameter of the `reset` command), there are six different reset cases:

- Jumper setting: **in** (default MDR configuration)
  - Pushing the reset button on the front panel will reset the MDR system but not the other boards present on the same VME backplane,
  - Executing the '`ResetServer(MDR_RESET_VME)`' function from the MDR Shell will have no effect at all on the MDR Server(s),
  - Executing the '`ResetServer(MDR_RESET_I960)`' function from the MDR Shell will reset the MDR system but not the other boards present on the same VME backplane,
- Jumper setting: **out**
  - Pushing the reset button on the front panel will reset the MDR system and the other boards present on the same VME backplane,
  - Executing the '`ResetServer(MDR_RESET_VME)`' command from the MDR Shell will have exactly the same effect as pushing the front panel reset button,
  - Executing the '`ResetServer(MDR_RESET_I960)`' command from the MDR Shell will reset the MDR system but not the other boards present on the same VME backplane.

Scan

Declaration:

```
STATUS Scan (DISK_INFO       * DiskInfoTbl,
             UINT              DiskInfoTblSize,
             UINT              * DisksFound,
             DISK_GROUP_INFO * DiskGroupInfoTbl,
             UINT              DiskGroupInfoTblSize,
             UINT              * DiskGroupsFound,
             UINT              Options);
```

Description: Scan for new disk and/or disk groups.

This function scans for new Fibre Channel devices: disks and/or disk groups and returns the gathered information in the `DiskInfoTbl`, `DisksFound`, `DiskGroupInfoTbl` and `DiskGroupsFound` parameters.

The `Options` parameter specifies which MDR actor (Client and/or Server) should perform the scan operation:

- `MDR_DISK_GROUP_SCAN_CLIENT`: Only the MDR Client shall scan for disks and disk groups.
- `MDR_DISK_GROUP_SCAN_SERVER`: Only the MDR Server shall scan for disks and disk groups. Currently, only a few features of the disk group are implemented in the MDR Server. Therefore, only one disk group can be found (with no name).

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SelfTest

Declaration:

```
STATUS SelfTest (char * Command);
```

Description: Check the data integrity of the MDR disk subsystem.

Checks the data integrity of the MDR disks area specified by the user.

The `Command` parameter represents the MDR Shell `selftest` command. For more information, please refer to the MDR Shell command.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

SetRecordingInfo

Declaration:

```
STATUS SetRecordingInfo (RECORDING * Recording);
```

Description: Set information to a recording.

This function requests to set the current recording information as specified in the `RECORDING` structure pointed to by `Recording`.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

SetRecTableInfo

Declaration:

```
STATUS SetRecTableInfo (REC_TABLE_INFO * RecTableInfo);
```

Description: Set the recording table information.

This function sets the recording table information as specified in the `REC_TABLE_INFO` structure pointed to by the `RecTableInfo` parameter.

Not all parameters in the `REC_TABLE_INFO` can be changed. The parameter `CurrentRecording` can be changed so that the current recording is set to any of the recordings present in the recording table.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

StartTransfer

Declaration:

```
STATUS StartTransfer (MDR_TRANSFER_ID * TransferId = NULL);
```

Description: Start a transfer.

This function requests the MDR server to start the transfer specified by `TransferId`.

Returned value: OK or ERROR.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

StopTransfer

Declaration:

```
STATUS StopTransfer (MDR_TRANSFER_ID * TransferId = NULL);
```

Description: Stop a transfer.

This function requests the MDR server to stop the transfer specified by `TransferId`.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

The end of the recording data may contain garbage data. The user has the responsibility to determine which part of the recording data is valid or not.

WaitTransfer

Declaration:

```
STATUS WaitTransfer (MDR_TRANSFER_ID * TransferId = NULL);
```

Description: Wait for a transfer to finish.

This function waits for the transfer specified by `TransferId` to finish. This function should be called after `StartTransfer` if the program needs to wait for the end of the transfer.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

WriteRecorderBlock

Declaration:

```
STATUS WriteRecorderBlock (UINT   FirstRecorderBlockToWrite,
                           void * DataBuffer,
                           UINT   DataBufferSize);
```

Description: Write recorder block(s) to the storage device.

This function writes one or more recorder blocks to the storage device. If the recording data (located in the recorder blocks) contains tags then these tags are overwritten according to the data contained inside the data buffer.

The `FirstRecorderBlockToWrite` parameter specifies the first recorder block to be written.

Returned value: `OK` or `ERROR`.

If `ERROR` is returned then the `errno` variable will be set accordingly. See `MdrDefs.h` for MDR `errno` values.

Using the `WriteRecorderBlock` function while making a recording affects the recording performance. The MDR Server may stop recording due to this.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WriteRecordingBlock

Declaration:

```
STATUS WriteRecordingBlock (UINT   RecordingNo,
                            UINT   StartRecordingBlock,
                            void * DataBuffer,
                            UINT   DataBufferSize);
```

Description: Write user block(s) of a recording.

This function writes one or more user blocks of the recording specified by RecordingNo.

If the recording contains tags then these tags are **not** modified by the MDR. The StartRecordingBlock parameter maps to the user block parameter in the RECORDING structure corresponding to the specified recording.

Returned value: OK or ERROR.

If ERROR is returned then the errno variable will be set accordingly. See MdrDefs.h for MDR errno values.

Using the WriteRecordingBlock function while making a recording affects the recording performance. The MDR Server may stop recording due to this.

8.4 MDR API Example Programs

Several example programs can be found in the mdr-host-sw/doc/ApiExamples directory. These programs illustrate the use of the MDR API.

The first of these programs is called MdrApiDemo. The MDR API example program performs some basic operations:

- Create an MDR Client object
- Create some recordings
- Record data
- Rename a recording

This scenario may be used as a first communication test between a user's program and the MDR Server. All the structures needed inside the API functions are allocated in this program. In fact, all the API functions check the input parameter(s) in order to avoid some unexpected and fatal problems but never try to correct the input parameters if those are wrong. Typically, if one empty pointer is passed as an input pointer, the API function will exit returning ERROR and an error number (stored in the errno variable).

The second program is called MdrDumpRec and is used to dump (or print) the data of a recording to the output (console by default). On UNIX or Windows platforms for instance, this program can be used to store recording data to any kind of device by using UNIX pipes and/or redirections. The command line is the following:

```
MdrDumpRec <RecNb> <ServerName> <DiskGroupName> [<FirstRecBlkToDump>]
  [<LastRecBlkToDump>]
```

Where:

<RecNb> is the number (or name) of the recording to be dumped,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

<ServerName> is the name of the MDR Server to try to connect to. This parameter shall be provided by the user. The server name NULL is used to indicate to the program that it should not try to connect to an MDR Server.
<DiskGroupName> is the name of the disk group to try to connect to. This parameter shall be provided by the user. The disk group name NULL is used to indicate to the program that it should not try to connect to a disk group.
<FirstRecBlkToDump> is the first user block of the recording to be dumped,
<LastRecBlkToDump> is the last user block of the recording to be dumped

Examples:

- `MdrDumpRec 1 mdr-250 NULL | dd of=/dev/rmt/0 bs=1m`

copy the recording 1 to the tape (located in /dev/rmt/0) via the MDR server mdr-250.

- `MdrDumpRec 1 NULL MyDiskGroup | dd of=/dev/rmt/0 bs=1m`

copy the recording 1 to the tape (located in /dev/rmt/0) via the direct FC connection. The recording is located in the disk group called `MyDiskGroup`.

- `MdrDumpRec 1 mdr-250 NULL 0 15 > Recording_1`

copy the recording 1 (user blocks 0 to 15 only) to the file Recording_1.

Note: It is possible to read data back from tape to the workstation using the UNIX `dd` command: `dd if=/dev/rmt/0 of=Recording_1 bs=1m`

8.5 MDR API Limitation

In order to use the 'direct access to storage device' API functions on Windows systems, the QLogic Driver has to be installed first.
The QLogic Driver must be installed on systems running Windows NT4.0 or 2000. Therefore, it is not possible to access the MDR storage device directly from hosts running Windows 95/98/Millenium.

8.6 Compilation Instructions

The MDR API library has been compiled with several compilers according to the platform:

- Windows platforms Microsoft Visual C++ 6.0
- Other platforms GNU compiler supplied by the FSF (Free Software Foundation). Details about the gnu compiler can be found at GNU's web site: http://gcc.gnu.org.

To compile your own program and link it with the MDR API library, do the following operations:

Find (and install if required) the compiler according to your platform. Currently, the MDR project has been compiled with the following compilers:

- `ppc-vxworks` GCC version 2.7.2-90126 (VxWorks 5.3.1)
- `sun-solaris` GCC version 2.95.2
- `x86-win32` Microsoft Visual C++ (Latest service pack)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If you platform is the Windows platform then open the workspace using Microsoft Visual C++. Otherwise, refer to the `mdr-host-sw/doc/ApiExamples/<type-of-platform>/Makefile` to compile your program (change the environment variable, directories and file names if needed).

Compile your program using the integrated build command (Windows platform) or `make install` command (other platforms).

Note: The MDR API is not thread-safe.

8.7 Externals Libraries

Some external libraries are required to compile the MDR API demo program or a project. Please refer to the project settings (Windows platform) or the `mdr-host-sw/doc/ApiExamples/<type-of-platform>/Makefile`, item `LIBS` (other platforms).

Note: When linking a program using the MDR API (`sun-solaris` platform), it is required to add the library `libdvui.a` if the Solaris VMEbus driver (provided by FORCE) has not been previously installed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9 MDR Operations

This chapter describes a subset of the MDR operations.
Please refer to the chapter 4: *MDR Shell Commands* and chapter 5: *MDR API Functions* for a detailed description of the MDR Shell commands and MDR API functions used in this chapter.

9.1 Disk Groups

A Disk Group is an MDR feature that allows grouping a set of Fibre Channel disks together. Each Disk Group on a JBOD (or RAID) is given a name upon creation, and is accessed as a single Fibre Channel disk during read or write operations.

Currently, a limited version of the disk group has been implemented on the MDR Server and the complete version of the disk group in the MDR Host Software (used while accessing directly the MDR disk subsystem via FC). The user should always first use the Host MDR software and FC Direct Connection to partition a JBOD into Disk Groups. However, if the MDR Server is used to create recordings on a set of disks before the user defines a disk group on those disks, then the user may do the following to safely access the recordings:
- Establish an FC Direct Connection between host computer and JBOD
- Set Host mdr.ini parameter `Disks=<disk-group-name>` (i.e., `Disks=vmetro-1`)
- Run the Host MDR Shell and create disk group 'vmetro-1', which includes the disks where the recordings reside.
- When asked if you want to clear the Recording Table on the disks, choose 'n'.
The MDR Shell can now access all recordings within disk group 'vmetro-1'.

When using the MDR Shell on an MDR board, the command `connect-disk-group` is available. In this case, the command functions the same as the Host-based MDR Shell when no Direct Connection is present. Because this command is sent to the MDR Server (not local MDR API), the command can only use a list of FC port Ids (i.e., `connect 8-11`). When using Disk Groups from a Host MDR Shell on a system with a FC Direct Connection, then the Disk Group name must be used. The Disk Group name is converted by the MDR API into a list of disk port Ids before being sent to the MDR Server.

9.1.1 Disk Group

The main disk group features are:

- Grouping a set of FC disks into a disk group. Therefore, the group of disks composing an MDR disk group is then used as one FC disk by the user via the MDR API. Note that a disk may be a member of only one Disk Group.
- Striping of the data over all the disks composing a disk group. This allows high-speed read/write transfers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.1.2 Disk Group in MDR Host Software

As described above, a disk group is an entity containing one or several FC disks. In order to be easily manipulated, each disk group shall be named when created. Later, the user can use this name to manage and access the data (read/write) on the disk group.
The disk group includes the following features:

- Management: via the MDR API, the user can create and destroy disk groups, add and remove disks from a disk group.
- Scan: via the MDR API, the user can find the available disks. Because the disk group information is stored on the disks itself, a scan operation can also find information about the existing disk groups.

Note:
For more information on the MDR API functions, please refer to Chapter 8: *MDR API Functions* and to the MDR API example programs.
For more information on the configuration items relative to the disk group, please refer to the Chapter 10:*Configuration Files*.

9.1.3 Disk Group in MDR Server

Currently, the disk group implemented in the MDR Server is a partial version of the disk group implemented in the MDR Host Software.
It does not include:

- Saving of the disk group information to the disks included in the disk group
- Part of the management: The user cannot create or destroy disk groups, add or remove disks from a disk group. The user has only the possibility of connecting to a disk group (using the port id list of the disks to be included in the disk group) and disconnecting from a disk group.
- Disk group scan operation (as no information is saved on the disks)

In order to avoid these limitations, the user has the possibility to connect the MDR Client to the MDR Server **AND** use a direct FC link (via VMFC-2X00(P) board) to the FC disks. In this case, all the disk group operations (management, scan, read/write, etc) are realized via the direct FC connection, translated for the MDR Server if required and sent to the MDR Server.

9.2 Transfer Operations

9.2.1Transfer Types

Currently, the MDR includes two transfer types:

- Record — A data source is sending data to the MDR via an input device (DPIO input module). The MDR records the received data to the MDR disk subsystem (JBOD/RAID) via a VMFC2x00 module. The collection of recorded data is called a recording
- Playback — Recorded data (inside a recording) is read from the MDR disk subsystem by the MDR and sent to a receiver via an output device (DPIO output module)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Two MDR Shell commands are associated to these two operations:

- **record** prepares and start a record operation
- **playback** prepares and start a playback operation

9.2.2 Wait & No Wait Transfer Modes

The modes related to the transfer are:

- Standard mode The recording is recorded/played back only once (NonInf)
- Infinite mode The recording is recorded/played back infinitely (Inf)

9.2.2.1 Wait (WAIT) Mode

The first mode, WAIT, can be selected using the following commands:

- **record WAIT** Prepare, start a record operation and wait for the MDR system to complete the transfer
- **playback WAIT** Prepare, start a playback operation and wait for the MDR system to complete the playback

9.2.2.2 No Wait (NOWAIT) Mode

The second mode, NOWAIT, can be selected using the following commands:

- **record NOWAIT** Prepare, start a record operation and do **not** wait for the MDR system to complete the record
- **playback NOWAIT** Prepare, start a playback operation and do **not** wait for the MDR system to complete the playback

9.2.2.3 Notes

The default parameter depends on whether the command was issued as a standard or infinite mode as follows.

```
record == record NONINF == record WAIT NONINF
record INF == record NOWAIT INF
playback == playback NONINF == playback WAIT NONINF
playback INF == playback NOWAIT INF
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.2.3 Infinite & Standard Transfer Modes

The modes related to the transfer are:

- Standard mode — The recording is recorded/played back only once (NonInf)
- Infinite mode — The recording is recorded/played back infinitely (Inf)

9.2.3.1 Standard (Non-Infinite) Mode

In the standard or Non-Infinite mode, a record/playback operation behaves as follows:

- **`record NONINF`** — Prepare, start a record operation from the first block of the recording (on the MDR disk subsystem) in the standard mode and wait for the MDR system to complete the record. The record operation stops (if not interrupted by the user and without any time out or a power loss) when reaching the last block of the recording.
- **`playback NONINF`** — Prepare, start a playback operation from the first block of the recording (from the MDR disk subsystem) in the standard mode and wait for the MDR system to complete the playback. The playback operation stops (if not interrupted by the user and without any time out or a power loss) when reaching the last block of the recording.

9.2.3.2 Infinite Mode

In the Infinite mode, a record/playback operation behaves as follows:

- **`record INF`** — Prepare, start a record operation from the first block of the recording (on the MDR disk subsystem) in infinite mode. The record operation does **not** stop (unless interrupted by the user, a time out, or a power loss) when reaching the last block of the recording, but loops back to the first block.
- **`playback INF`** — Prepare, start a playback operation from the first block of the recording (on the MDR disk subsystem) in infinite mode. The playback operation does **not** stop (unless interrupted by the user, a time out, or a power loss) when reaching the last block of the recording, but loops back to the first block.

9.2.3.3 Notes

- When recording data in the infinite mode, the recording size must be a multiple of the recorder block size. If not, the recording size will be modified by the MDR before recording data
- The default parameter is `NONINF`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.2.4MDR API

As for the transfer type, the transfer mode is selected when calling the `PrepareTransfer` function in the MDR API. The field `Infinite` of the `MDR_TRANSFER` structure is used to specify the transfer mode.

At API level, a record or playback are transfers. Before being started, a transfer needs to be prepared using the function:

```
STATUS PrepareTransfer (MDR_TRANSFER    * MdrTransfer,
                        MDR_TRANSFER_ID * TransferId);
```

Where `MdrTransfer` is a structure describing the transfer to be carried out and `TransferId` the transfer identification number used to identify the transfer in future function calls. The field `TransferType` of the `MDR_TRANSFER` structure is used to specify the transfer type.

When a transfer is prepared, it can be started using the function:

```
STATUS StartTransfer (MDR_TRANSFER_ID * TransferId);
```

And stopped using the function:

```
STATUS StopTransfer (MDR_TRANSFER_ID * TransferId);
```

The user's program can also wait for the transfer to complete using the following function:

```
STATUS WaitTransfer (MDR_TRANSFER_ID * TransferId);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.3 Recovery from power failure

9.3.1 Power Failure

If a power failure occurs during a record operation, MDR provides a functionality to recover recording data. The MDR system is able to recover the recording data saved to the MDR disk subsystem if tags are inserted into the recording (insertion of tag is covered in a later paragraph).
The detection of power failure and the data recovery are carried out automatically during the boot sequence of the MDR:

- If the MDR detects that a record operation was carried out **without using** tag insertion then, in order to save any possibly recorded data, the recording is set as stored (and recorded) by the MDR.

In this case, the last part of the recording may contain valid data and/or garbage data. The user shall determine the valid recording data.

- If the MDR detects that a record operation was carried out **using** tag insertion then the following operations are executed:
  1. The number of valid recorded blocks is computed (using the tag information inserted inside the recording data),
  2. The recording is resized according to the number of recorded blocks found. Therefore, the recording contains **only** valid data (no garbage data). The rest of the data on the MDR disk subsystem may be retrieved using the `ReadRecorderBlock` function,
  3. The recording is set as `stored` and `recorded`.

9.3.2 Tag Insertion

The MDR system is able to insert tags inside the recording data during a record operation (in both standard and infinite mode).
Each tag is a 512-byte structure containing:

- The MDR identification value (also called magic number),
- The recorder block number,
- The validity of the recorder block,
- Reserved space.

There are two different ways of enabling the tag insertion for a recording:

- During the creation of a recording,
- After the creation of a recording.

The space used by the recording tags is added to the recording user size (not allocated inside the recording user size). Therefore, more recorder blocks may be required for a recording with tags (compared to the same recording without tags).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9.3.2.1.1 MDR Shell

As described in the chapter related to the MDR Shell commands, it is possible to create a recording using the following command:

```
new-recording [<Blocks> [<BlockSize>]] [<InsertTag>]
```

The optional parameter `<InsertTag>` can be used to enable the tag insertion to the recording data. For instance,

```
new-recording 1 G TAG
```

creates a 1 gigabyte recording with space for a recording tag in every recorder block (more information about the number of recorder blocks and the recorder block size can be found when using the `show` command).

Note: By default, a recording is created with the tag insertion **disabled**.
The command `tag-enable` is used to enable the tag insertion for a recording previously created without tags. Symmetrically, the command `tag-disable` is used to disable the tag insertion for a recording.

9.3.2.1.2 MDR API

Like the MDR Shell, the MDR API also allows to enable the tag insertion during the creation of a recording and to enable/disable the tag insertion for a recording afterwards.

As described in the chapter related to the MDR API functions, `CreateNewRecording` is used to create a new recording. The recording field `TagInfo` specifies the recording tag information. Currently, tag insertion is enabled when the `TagInfo` field is set to `1` and disabled when set to `0`.

The MDR API does not include any specific function to enable/disable the tag insertion for a recording. The `SetRecordingInfo` function should be used for this purpose:

- Get the information of the recording to modify the tag information field (using the `GetRecordingInfo` function),
- Modify the recording field `TagInfo` (from `0` to `1` or `1` to `0`),
- Set back the recording information (using the `SetRecordingInfo` function).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10 Configuration Files

The MDR Server and Shell programs, as well as the VxWorks operating system need some parameters to indicate how to configure the MDR during initialization. In the MIDAS FLASH memory, these parameters (also called items) are stored into files and are sorted into different sections noted between square brackets.

The files that can be configured are the following:

- **`mdr.ini`** MDR-specific configuration parameters file,
- **`vxbsp.ini`** VxWorks BSP configuration file,
- **`vxbp.txt`** VxWorks boot parameters file,
- **`mmon.ini`** MIDAS Monitor configuration file.

In the configurations files `mmon.ini`, `vxpsb.ini` and `mdr.ini`, the items that need to be set up already have a correct value when the MDR system is shipped. The user may change these items and consequently the behavior of the entire MDR system.

The file `vxbp.txt` need to be modified by the user in order for the MDR system to adapt its configuration (with network connection, standalone, acting as a gateway, etc).

Do not change these items unless you are sure of what you are doing!

In the following item list, any item related to a PMC slot should be set up according to the following values:

- 1 or 6 for MIDAS-220S
- 1 to 5 for MIDAS-250.

Note:

The `mmon.ini` file included in the software distribution is an example only! It should never be uploaded to the MDR except in the case of recovery from catastrophic failure. Even then its contents MUST be tailored to match the MDR unit for which it is destined. Likewise, the `mdr.ini` file included in the software distribution should be regarded merely as a template. It only reflects one model of MDR and should be tailored to your actual model before uploading. Again, this file should be uploaded only if absolutely necessary (major upgrade, recovery from catastrophic failure, etc.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.1 Summary Table

Configuration file	Section	Item	Used by				Modification level	Severity level
			MIDAS Monitor	VxWorks	MDR Server	MDR Shell		
mmon.ini	BoardInfo	Model	X				VMETRO	High
		EcoLevel	X				VMETRO	High
		TotalDramSize	X				VMETRO	High
		SerialNo	X				VMETRO	High
	BootInfo	TestUart	X				VMETRO	High
		UartLbBytes	X				VMETRO	High
		TestDram	X				VMETRO	High
		DramBankOffset	X				VMETRO	High
		ProbeOther	X				VMETRO	High
		ProbeUniverse	X				VMETRO	High
		TestUniverse	X				VMETRO	High
		ProbePxb	X				VMETRO	High
		ProbePmc	X				VMETRO	High
	AutoStart	ProcessPriority	X				VMETRO	High
		StartAddr	X				VMETRO	High
		WaitTime	X				VMETRO	High
vxbsp.ini	SysMemCfg	SysPhysMemSize		X			User / VMETRO	High
	MDC	PmcSlot		X			User / VMETRO	High
		EthSpeed		X			User	Low
	VmeInterface	VmeA32SlaveDisable		X			User / VMETRO	High
		VmeA32SlaveBase		X			User / VMETRO	High
		VmeA32Master2Base		X			User / VMETRO	High
	StartUp	fncX		X			User / VMETRO	High
		fncXargY		X			User / VMETRO	High
	SCSI	SCSIUnitXPmcSlot		X			User	Medium
		SCSIUnitXPciBaseAdrs		X			User	Medium
mdr.ini	MainInfo	MdrModelName			X		User / VMETRO	High
		Verbose			X	X	User	Low
		SharedMemoryDataVmeBaseAdrs			X	X	User	Medium
	StorageInfo	Disks			X	X	User	Low
		VMFC-2100UnitXPmcSlot			X		User	Low
		VMFC-2100UnitXPortId			X		User	Low
		DiskTransferBlockSize			X	X	User	Low
		FcLoopRetryTime			X	X	User	Low
		UseHostFcAdapter				X	User	Low
	DpioInfo	SuspendSignal			X		User	Low
		DPIO-InputUnits			X		User	Low
		DPIO-OutputUnits			X		User	Low
		DPIO-InputUnitXPmcSlot			X		User	Low
		DPIO-OutputUnitXPmcSlot			X		User	Low
		StartInputWaitTime			X		User	Low
		InputWaitTime			X		User	Low
		OutputWaitTime			X		User	Low
		ByteSwap			X		User	Low
		OutputFreqDivisor			X		User	Low
		WaitForSync			X		User	Low
		PeclStrobe			X		User	Low
		InputStartSignal			X		User	Low
		OutputStartSignal			X		User	Low
		TransferOptions			X		User	Low
	Servers	sX			X		User	Low
	SCSI	ScsiTapeBusId			X		User	Low

Modification level:

- **User** User, by his own initiative, can change the item,
- **User/VMETRO** User (under the VMETRO control) can change the item,
- **VMETRO** Only VMETRO should change the item.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Severity level:

Low	Modify the behavior of the program/operating system without major consequences,
Medium	Item which can modify the program/operating system with important consequences,
High	Item which have very high consequences on the behavior of the related program/operating system.

10.2 Item modification

An item can be modified in one of the configuration files using the MIDAS Monitor functions `mgf` or `iset`. For further information on the use of the `mgf` function, see section *7.7 Configuring an MDR Unit by preloading the Boot Parameter Files.*

The `iset` command can be used as follows:

Following the board's self-test, the MDR unit will present a short countdown to permit you to enter the MIDAS monitor:

Press any key to interrupt the countdown and then press the Enter key 2 or 3 times within 2 seconds (to perform autobaud). The MIDAS monitor prompt # should then appear.

Update an item with the `iset` command:

```
#iset <section> <item> <text> <file>
```

For instance, to change the number of disks used (to 8) by the MDR system, type in the MIDAS Monitor:

```
#iset StorageInfo Disks 8 mdr.ini
```

Exit the monitor with the `quit` command:

```
#quit
```

The MDR unit will reboot. This time, allow the countdown to expire and the MDR firmware will begin executing.
For more information on the `iset` command, please see the MIDAS Monitor online help (`help` command under the MIDAS Monitor).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.3 MDR system configuration file (mdr.ini)

Below is a list of the parameters found in **mdr.ini**.

[MainInfo]

This section contains the main information the MDR Server needs to know.

MdrModelName

Model name of the MDR system. The valid model names are MDR-250 (MIDAS-250 board), MDR-220 (MIDAS-220 board). This item is required by the MDR Server and must be defined.

SharedMemoryDataVmeBaseAdrs

This item specifies the VME base address of the Shared Memory Data structure. For further information on the supported PMCs, see *Appendix F: Supported PMCs Documentation*.

If the MDR Server cannot find the SharedMemoryDataVmeBaseAdrs item, then no structure will be mapped to the VMEbus during a recording. This mode is then disabled.
If the SharedMemoryDataVmeBaseAdrs item equals to a value that is located in an existing VME Universe slave image, then the MDR Server will display an error message during the initialization phase and the VME mapping will be disabled.
The shared memory data contains some information on the MDR Server program (signature, version numbers and MDR model name) and some addresses which indicate where to read the monitor and control data structures on the VMEbus. Therefore, during a recording, reading the SHARED_MEMORY_HEADER structure on the VMEbus at the SharedMemoryDataVmeBaseAdrs will provide the recording progress indication data.
For example, to read this structure on the VMEbus at the address 0x06000000, set up the SharedMemoryDataVmeBaseAdrs to 0x06000000.

Verbose

This item defines the verbose level for the MDR Shell and can have the following values specifying how detailed error messages should be printed out:

- NORMAL: Normal mode. One single line of error message every time a command fails.
- DETAILED: Detailed mode. Prints out more detailed information about the error. This error message should clearly identify the problem/error.
- DEBUG: Debug mode. Prints out all possible error messages in order to identify (and trace) the problem. This mode should be used only when debugging.
  If the Verbose item is omitted from the mdr.ini file then the verbose level is set by default to NORMAL.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[StorageInfo]

This section contains information the MDR programs need to know about the storage device connected to the MDR and its connection (PMC boards, initialization time, etc)

Disks

MDR Server: List of disk port identifiers to be controlled by the MDR. If this item is set to 0 or is not defined then the disk port id used by default is 8.

MDR Host Software: Name of the disk group to be controlled by the MDR Host Software. This item does not have any default value.

DiskTransferBlockSize

Size, in KiB, of each transfer from/to the disks. In order to improve the MDR performances, 512 KiB is typically used when the disk subsystem is an 8-disk JBOD and 4096 KiB when it is a CIPRICO RAID.

VMFC-2100UnitXPmcSlot

PMC slot of VMFC-2100 unit X. If this item is set to 0 or undefined (whereas X is in the range 1-VMFC-2100Units) then the MDR Server exits returning an error.

VMFC-2100UnitXPortId

Port ID number of the VMFC-2100 unit X, where X is a decimal number in the range 1-125. If this item is set undefined (whereas X is in the range 0-125) then the MDR Server exits returning an error.

Also, the VMFC-2100 Port Id should be less than any disk Port id.

FcLoopRetryTime

Time, in seconds, the MDR server should try (and retry) to establish a connection with the MDR disk subsystem. This time includes the time needed to initialize the SCSI controller(s) and the time to detect (inquiry) the entire number of hard drives (this number is specified by the Disks item in this section).

If this item is set to 0 then the MDR server will try forever to connect to the MDR disk subsystem. If it is undefined then the default value 120 (seconds) is used.

UseHostFcAdapter

Indicates that a disk subsystem if present through a VMFC-2X00(P) board. This item is taken into account by the MDR Shell program on the host workstation. Currently, only the x86-win32 and sun-solaris platforms support this item.

If this item is set up to 1 then the MDR Shell will try to connect to the disk subsystem. If this item is set up to 0 or missing then MDR Shell will not try to connect to the MDR disk subsystem.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[DpioInfo]

This section contains information the MDR Server needs to know about the DPIO modules installed on the MDR.

SuspendSignal

`Disable` or `Enable` DPIO suspend signal. If this item is not defined then the default value is `Enable`.

DPIO-InputUnits

Number of DPIO input units that are used. Currently, this value is limited to 0 (no DPIO input PMC installed) or 1. If this item is not defined then the MDR Server ignores the DPIO input modules and the items `DPIO-InputUnitXPmcSlot` presented below.

DPIO-OutputUnits

Number of DPIO output units that are used. Currently, this value is limited to 0 (no DPIO output PMC installed) or 1. If this item is not defined then the MDR Server ignores the DPIO input modules and the following items `DPIO-OutputUnitXPmcSlot` presented below.

DPIO-InputUnitXPmcSlot

PMC slot of DPIO input unit X. This value should be 1 (or 2 according to the MDR standard configuration) for performance reasons. If the `DPIO-InputUnits` item is set to 0 or not defined then this item is ignored. If this item is missing then the DPIO input module X is ignored.

DPIO-OutputUnitXPmcSlot

PMC slot of DPIO output unit X, where X is a decimal number in the range 1-6. This value should be 1 or 2 for performance reasons. If the `DPIO-OutputUnits` item is set to 0 or not defined then this item is not used. If this item is missing then the DPIO output module X is ignored.

StartInputWaitTime

`Maximum` time the MDR server will wait before receiving the first recorder block of data. The value of this item could be considered as the maximum time that the user could use to launch the DPIO input stream. If this item is missing or set to –1 then the MDR Server will wait forever at the beginning of the recording and never return, provided there is no input data (the watchdog timer is disabled in this case).

InputWaitTime

Maximum time the MDR server will wait between two consecutive recorder blocks of input data. The value of this item could be considered as the maximum time the data stream can use to transfer a block of data in the recorder (its size is the RecorderBlockSize value). If this item is missing or set to –1 then the MDR server will wait forever during the recording and never return, provided there is no input data (the watchdog timer is disabled in this case).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The values of `StartInputWaitTime` and `InputWaitTime`, divided by 60, represent the number of seconds the MDR server will wait before stopping the current record operation (if there isn't any input stream).
This mode enables a user-defined watchdog timer for the DPIO data input stream: if there is no data for a period of time then the MDR server assumes that the end of recording has been reached. It will then stop the recording in progress and the DPIO input module, set up the new recording size (according to the number of blocks received) and return to the MDR host software with an error message. If no data at all is received then the recording size is set to the `RecordingBlockSize` value.

`OutputWaitTime`

Maximum time the MDR server will wait between two consecutive recorder blocks while playing back data. The value of this item could be considered as the maximum time the data stream can use to transfer a block of data from the recorder (its size is the RecorderBlockSize value). If this item is missing or set to –1 then the MDR server will wait forever during the playback and never return, provided that the receiver never receive the data (the watchdog timer is disabled in this case).
As for `StartInputWaitTime` and `InputWaitTime`, `OutputWaitTime`, divided by 60, represent the number of seconds the MDR server will wait before stopping the current playback operation.

`ByteSwap`

Enables (if set to `Enable`) or disables (if set to `Disable`) the byte swapping. If this item is missing, then the byte swapping is disabled.

`OutputFreqDivisor`

Output oscillator frequency divisor. This item specified the value by which the frequency of the oscillator frequency (on an DPIO output module only) is divided. The allowed values for this item are 0, 2, 4 and 8. If this item is set to any other value or is missing then the default value (0) is used.

`WaitForSync`

Enables (if set to `Enable`) or disables (if set to `Disable`) the wait for SYNC functionality of the DPIO device. If this item is missing then the SYNC signal is not used (Wait for sync functionality is disabled).

`PeclStrobe`

Enables (if set to `Enable`) or disables (if set to `Disable`) the PECL strobe functionality of the DPIO device. If this item is missing then the PECL strobe is disabled.

`InputStartSignal`

Select the start signal the data source is triggered on when starting a record operation. The PIO1 (resp. PIO2) signal is used if this item is set to `PIO1` (resp. `PIO2`). The value `Disable` is used to disable this feature: in this case, the MDR do not trigger the data source.
If this item is missing then the PIO1 signal is used by the MDR to trigger the data source.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OutputStartSignal

Select the start signal the data receiver uses to trigger the MDR when starting a playback operation. The `PIO1` (resp. `PIO2`) signal is used if this item is set to `PIO1` (resp. `PIO2`). The value `Disable` is used to disable this feature: in this case, the MDR do not trigger the data source.

If this item is missing then the MDR is not triggered by the data receiver and will start sending data as soon as the playback command is received.

TransferOptions

Select the transfer options. The transfer option value can be expressed either in decimal or hexadecimal and is built using a bit mask:

- Bit 0: DPIO DMA Pause/Resume.
  Set this bit to `1`(`0`) to enable (disable) the DMA Pause/Resume mechanism used when recording data. This mechanism allows the FPDP SUSPEND signal to be used to signal when all the memory buffers used by the MDR are full and when the source should pause the data transfer.
  This option is only valid when the `SuspendSignal` item is set to `Enable`.

[Servers]

This section contains information the MDR Shell needs to know about the servers it should try to connect to.

sX

Hostname of server X (where X is a decimal number).This parameter is only needed by the MDR Shell program. If any `sX` items are defined then the MDR Shell will prompt the user for the MDR Server hostname.

[SCSI]

This section contains information the MDR Server needs to know about the SCSI modules installed on the MDR.

ScsiTapeBusId

Bus identification number used by the tape drive connected to the SCSI PMC board. This number is a decimal number between 0 and 15 except 7 (this Id is used by the SCSI PMC controller). If this item is not present then 0 is used as default bus ID.

10.4 VxWorks configuration file (vxbsp.ini)

Below is a list of the **`vxbsp.ini`** parameters that are relevant to the MDR configuration:

[SysMemCfg]

This section contains information the BSP needs to know about the memory configuration of the system on the MIDAS board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SysPhysMemSize

Size of memory that VxWorks should handle. The current value used is `0x1b00000` for MDR systems and the user should not be change this item on its own initiative.

[MDC]

This section contains the information the BSP needs to know about the Midas DC (name of the driver for the DE520 PMC board) installed on the MIDAS board.

PmcSlot

PMC slot where the Ethernet board is installed. Currently, this parameter is optional and overrides the boot device <slot> number (for more information, see *Chapter 3 MDR Host Software.*

EthSpeed

Speed of the Ethernet network.

Values currently supported are:

10Mb	10 Mb/s standard Ethernet (default value)
100Mb	100 Mb/s fast Ethernet
AutoSense	Auto-sense and switch to the correct speed

Note:

Some MDR-250 units may be shipped with an Ethernet PMC manufactured by *Osicom*. At present, these PMCs do not correctly auto-sense the Ethernet carrier speed, so the `AutoSense` option must not be used. `EthSpeed` must be set to `10Mb` or `100Mb` as appropriate for the network to which the MDR will be connected.

[VmeInterface]

This section contains information the BSP needs to know about the VME interface on the MIDAS board.

VmeA32SlaveDisable

VME A32 slave image enabled flag. If this item is set to 1 (respectively 0) then the VME A32 slave image is disabled (respectively enable). If this item is not set correctly or missing then the default value is 0.

This parameter should be defined to 0 (or missing) when the network interface is `sm0` (SM Network requires a VME A32 slave image).

VmeA32SlaveBase

Base address of where the MIDAS DRAM should be mapped in VME A32. If this item is missing then the VME A32 slave base address is computed by the BSP.

The default BSP computation puts the VME A32 slave base of a MIDAS acting as an SM master (with a processor number of 0) at address 0x0.

For processor numbers different from 0, the following formula is used:

```
VmeBase = (ProcNum * 0x10000) + 0x4000000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VmeA32Master2Base

Base address (in VME A32 space) of an optional secondary VME master window the i960 processor of the MIDAS board should set. This master window, and therefore this item, is required sometimes on the MIDAS side in order to be able to communicate with some SBCs in SM (For instance, the Power PC boards MV230X/260X).

[StartUp]

This section contains information the BSP needs to know about the functions to call after booting.

fncX

Function to be automatically called after booting VxWorks on the MIDAS.

fncXargY

Argument to function X.

The `startUp` section should have the following items:

```
fnc1=_sp
fnc1arg1=_MdrServer__Fv
```

[SCSI]

This section contains information the BSP needs to know about the SCSI PMCs installed on the MIDAS board.

SCSIUnitXPmcSlot

PMC slot where the SCSI PMC module number X is installed. X is a decimal number starting at 1 (2, 3, 4…). If no items are specified then the MIDAS BSP will not try to detect SCSI PMCs.

In the MDR application, this item is used currently to connect a SCSI tape drive. It should be setup as reference in the MDR standard configurations.

SCSIUnitXPciBaseAdrs

PCI base address where the driver of the SCSI PMC module number X is installed.

The default value for this item is `0x40002000`. Do not change the value of this item unless you are sure of what you are doing!

10.5 VxWorks boot parameters file (vxbp.txt)

The **`vxbp.txt`** file is set up in VxWorks with the program called `bootChange`. In any case, the user shouldn't try to modify this file using any other program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.6 MIDAS Monitor configuration file (mmon.ini)

The **mmon.ini** file contains information on the MIDAS system and the behavior of the MIDAS Monitor during boot time. None of the parameters in this file need to be changed.
Below is a brief description of all the items relevant to the MDR configuration:

[BoardInfo]

This section contains information the MIDAS Monitor needs to know about the MIDAS board.

Model

MIDAS board model. The different model names currently supported and compatible with the MDR applications are:

M220S	MIDAS-220S
M250	MIDAS-250

EcoLevel

ECO level of the MIDAS board.

TotalDramSize

Total size (in MiB) of the DRAM installed on the MIDAS board. This value can be read on the MIDAS board itself.

SerialNo

MIDAS Board serial number. This number can be read on the MIDAS board itself.

[BootInfo]

This section contains information the MIDAS Monitor needs to know how to boot.

TestUart

UART test indicator. The MIDAS Monitor tests the UART if this item is set to 1, doesn't test it if this item is set to 0. This item should be set to 0.

UartLbBytes

Number of bytes sent each time for the UART loop test performed by the MIDAS Monitor. This item is taken into account only when the `TestUart` item above is defined to 1. This item should be set to 10.

TestDram

DRAM test indicator. The MIDAS Monitor tests the DRAM if this item is set to 1, doesn't test it is this item is set to 0. This item should be set to 0.

DramBankOffset

DRAM address where the MIDAS Monitor DRAM test should begin. Thus item should be set to `0xf80000`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ProbeOther
This item should be set to 1.

ProbeUniverse
Universe probe indicator. The MIDAS Monitor probes the Universe chip if this item is set to 1, and does not probe it if set to 0. This item should be set to 1.

TestUniverse
Universe test indicator. The MIDAS Monitor tests the Universe chip if this item is set to 1, and does not test it if set to 0. This item should be set to 1.

ProbePxb
PXB probe indicator. The MIDAS Monitor probes the PXB if this item is set to 1, and does not probe it if set to 0.

ProbePmc
PMC probe indicator. If this item is set to `All` then the MIDAS Monitor probes all the PMCs installed on the MIDAS board.

[AutoStart]

This section contains information the MIDAS Monitor needs to know about the start up procedure.

ProcessPriority
This item should be set to `16`.

StartAddr
DRAM address where the MIDAS Monitor jumps after performing all the tests. This item should be set to `0xfe000000.`

WaitTime
MIDAS Monitor countdown time (in seconds) before jumping to `StartAddr`. If this item is missing then a default time will be used (5 seconds).

10.7 Configuring an MDR Unit by Preloading the Boot Parameter Files

If the existing boot parameters prohibit the system from initializing sufficiently for you to use the `bootChange` command, you may preload customized boot parameters through the MIDAS monitor using a terminal program which provides the XMODEM file transfer protocol.

The boot parameters are held in a FLASH ROM system file called `vxbp.txt`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example boot parameter files may be found in <path>/mdr-server/etc/vxbp*.txt:

- vxbp-min.txt is the minimum necessary to operate with SM Network as external network provided by a separate SBC (used in MDR-x20 units).
- vxbp-netif.txt is an example of boot parameters for a MIDAS serving as the network gateway. This may also be used for an MDR-250 unit with an Ethernet PMC.

You may modify these files to suit your system. The format of the string (which must be contained in a single line) is:

```
interface(unit,procnum)hostname:filepath keyword=value [...]
```

where interface is the name of the interface to use, unit is generally 0, procnum is the processor number.
The MDR is a standalone system, it doesn't need any external host to load a file from. Therefore, the user doesn't have to use the hostname and filepath fields.
The keyword is a 1- or 2-letter symbol shown in the bootChange examples above and the value is whatever is appropriate for your system.

vxbp-nonet.txt is used when no external network is required. vxbp-nonet.txt may also be used as a starting point to get the MDR unit up so that the bootChange command can be used.

To install vxbp-nonet.txt, use the following procedure.

1) Enter the Midas Monitor and delete vxbp.txt:
```
# midel vxbp.txt
```

2) On the Host computer, go to <path>/mdr/etc, and type 'dir' to get the size of file vxbp-nonet.txt.

3) Type the following at the Midas Monitor prompt:
```
# mgf vxbp.txt <size>
```
Note that, after mgf, you specify vxbp.txt and not vxbp-nonet.txt. This is because vxbp.txt is the name of the target file on the board, while vxbp-nonet.txt is the file you are copying from.

4) Within HyperTerminal, click on Transfer->Send, set Protocol=Xmodem and Browse your computer to fetch file <path>/mdr/etc/vxbp-nonet.txt.
Then click on Send.

5) When the transfer is done, type quit to reboot the board.
When the board is done rebooting, type mdrshell.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.7.1.1 Boot parameters preload

If it is necessary to preload boot parameters, use a terminal emulator which provides the XMODEM protocol to communicate with the MDR through its front-panel serial port; 9600 bps, 8 data bits, 1 stop bit no parity, no flow control.

Following the board's self-test, the MDR unit will present a short countdown to permit you to enter the MIDAS monitor:
Press any key to interrupt the countdown and then press the Enter key 2 or 3 times within 2 seconds (to perform autobaud). The MIDAS monitor prompt # should then appear.
Clear the old boot parameters with the `del` command:

```
#del vxbp.txt
```

This will take some time while the FLASH is updated.

Upload the new boot parameters with the `mgf` command:

```
#mgf vxbp.txt <size>
```

where `<size>` is the size of the desired boot-parameter file to be uploaded. The MIDAS monitor will begin an XMODEM receive. Instruct your terminal program to begin an XMODEM send of the file containing the boot parameters you wish to use. Once the file is sent, there will be a pause while the new `vxbp.txt` file is written to FLASH.

When the MIDAS Monitor prompt returns, check the contents of the file, use the `more` command:

```
#more vxbp.txt
```

Exit the monitor with the `quit` command:

```
#quit
```

The MDR unit will reboot. This time, allow the countdown to expire and the MDR firmware will begin executing.
You may also need to add or change some items related to your host network in order to communicate with the MDR. In particular, host tables and gateway information must be updated as appropriate. Consult your system administrator and/or your local network documentation for information and instructions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A Troubleshooting

Below is a table of the most frequently occurring error situations and the actions for correcting them.

Symptom/Error Message	Remedy
The MDR client (Shell or API) is unable to connect to the MDR server.	Connect a terminal to the MDR (i960) serial port, and check for error messages.
The MDR system reports that some disks could not be found.	Make sure that the Fibre Channel cable is properly attached between the MDR and JBOD/RAID. Check also that all the disks are properly inserted.
There is no message sent to the i960 serial port when the MDR is powered up.	Check that the MDR board is properly inserted into your VME crate. Contact VMETRO to verify that the board is correctly inserted.
The MDR client (Shell or API) is unable to connect to the MDR server and the latter does not print any error messages to the serial port.	Check that the network (boot) parameters are correctly set up.
The MDR server reports either DPIO FIFO overflow or MIDAS Memory buffer overflow during a recording.	Make sure that your data source is not sending data faster than the MDR system is able to handle. Also, check that the expected number of disks is being used by the MDR server.
The MDR system only uses X out of Y (expected) disks (with X<Y).	Check that the value of the item `Disks` in section `StorageInfo` in `mdr.ini` is set to X.
The MDR system reports that it is unable to configure a VMFC-2100 card.	Check that the VMFC-2100 card is positioned in the PMC slot specified by the item `VMFC-2100UnitXPmcSlot` (X is 3 or 4) in section `StorageInfo` in the `mdr.ini` file. Contact VMETRO to verify that the card is in the correct PMC slot.
FC Loop is : DOWN	Check that the Fibre Channel cable is correctly inserted into the JBOD/RAID and the VMFC-2X00. Power up the JBOD/RAID and reboot the system by pressing the Reset button on the MIDAS board. Check that there is proper air-cooling.
MDR client unable to connect to the server.	Check the `sx` parameters in the `mdr.ini` file.
Recording table is invalid: should be cleared.	If the recording data is not critical, clear the recording table (MDR Shell command `Clear`), Otherwise, try to swap disks in use (JBOD) if some of them have been previously swapped. If a new version has been installed on the MDR, call VMETRO for information on how to update the recording table without losing data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B Error Messages

Error Name	Error Number	Error Message	Error Description
S_mdr_COMMBUF_TOO_SMALL	0x55500001	The TCP/IP communication buffer is too small.	The MDR Server or the MDR Client is trying to send, through the communication buffer, a buffer larger than the communication buffer size.
S_mdr_COMM_OUT_OF_SYNCH	0x55500002	Communication out of synchronization.	The MDR Client has received a response packet that doesn't match the expected response to the command sent previously to the MDR Server.
S_mdr_COMM_INTERFACE_DOWN	0x55500003	Communication interface down.	The communication interface is down
S_mdr_CURRENT_RECORDING_NOT_SET	0x55500004	The current recording is not set.	There is no recording present in the recording table and the MDR software expects the current recording to be set.
S_mdr_DISK_ALREADY_IN_USE	0x55500005	Disk already in use	The disk is already included in a disk group. It cannot be included in two different disk groups at the same time.
S_mdr_DISK_ERROR	0x55500006	MDR disk error.	The MDR Server is unable to write to the storage device.
S_mdr_DISK_GROUP_NOT_FOUND	0x55500007	Disk group not found	The disk group cannot be found. The disk group name may be invalid or some disks contained in the disk group may be missing or unreachable.
S_mdr_DISK_GROUP_ALREADY_PRESENT	0x55500008	Disk group already present	The disk group with the same name is already present on the FC loops.
S_mdr_DPIO_EEPROM_DAMAGED	0x55500009	DPIO EEPROM damaged	The EEPROM of the DPIO is damaged.
S_mdr_DPIO_ERROR	0x5550000A	DPIO ERROR	DPIO error.
S_mdr_DPIO_NOT_PRESENT	0x5550000B	DPIO not present.	DPIO module is not installed in the MDR System and the MDR Server has received a command that requires the use of this DPIO module.
S_mdr_DPIO_OVERFLOW	0x5550000C	Overflow in FIFO of DPIO device.	The data source is sending data faster than what the DPIO input module is able to transfer. If the data path is FPDP and the data source supports the Suspend signal, then the configuration item `SuspendEnable` (section `DpioInfo` in the configuration file mdr.ini) should be set to 1 (i.e. enabled).
S_mdr_ILLEGAL_OPERATION	0x5550000D	Illegal operation.	The current state does not allow the command. This error is reported, for instance, when the MDR is recording and receives in the same time another command that involves a modification of the recording table.
S_mdr_INVALID_ARGUMENT	0x5550000E	Invalid argument.	At least one argument of the command is not valid.
S_mdr_INVALID_MEMORY_BLK_TBL	0x5550000F	Invalid Memory Block Table.	The MDR Memory Block Table has not been initialized correctly.
S_mdr_INVALID_MODEL_NAME	0x55500010	Invalid MDR model name.	The MDR model name specified in the configuration file `mdr.ini` is not valid.
S_mdr_INVALID_PARAMETER	0x55500011	Invalid parameter.	An MDR Server / MDR API / MDR Shell function parameter is not valid.
S_mdr_INVALID_RECORDER_BLK_TAG	0x55500012	Invalid recorder block tag	The tag present inside the data of the recorder block is invalid.
S_mdr_INVALID_RECORDING_NUMBER	0x55500013	Invalid recording number.	The recording number specified is not valid. The recording number must be a positive number, not null, and lower or equal to the number of the last recording present in the recording table.
S_mdr_INVALID_RECORDING_STATE	0x55500014	Invalid recording state	The state of the recording is invalid. For instance, the MDR Shell `copy disks host ./ 1` is executed whereas the recording number 1 is not stored.
S_mdr_INVALID_USER_BLKSIZE	0x55500015	Invalid user block size.	Reserved for future use.
S_mdr_MDR_INI_FILE_ERROR	0x55500016	MDR ini file error.	The MDR Server / MDR API / MDR Shell is unable to read an item (or its value) from the configuration file mdr.ini.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

S_mdr_MIDAS_MEMORY_OVERFLOW	0x55500017	MIDAS memory block overflow.	The data source is sending data faster than the MDR is able to write to the storage device. This error is different from the S_mdr_DPIO_OVERFLOW error: in this case, the DPIO input module is able to transfer all the data from the data source to the MDR Server but the MDR Server is unable to sustain the transfer rate when writing to the storage device.
S_mdr_MISMATCHING_PARAMETERS	0x55500018	Mismatch between parameters.	A specified parameter does not match an internal one.
S_mdr_MISMATCHING_VERSION_NB	0x55500019	Mismatch between MDR version numbers.	The MDR Server and MDR API (MDR Shell) version numbers are different. They must be equal. You must not use different software version for the MDR API and MDR Server.
S_mdr_NOT_CONNECTED_TO_SERVER	0x5550001A	Not connected to server.	The MDR Client is not connected to an MDR Server and a command that requires a connection is sent to the MDR API.
S_mdr_NOT_ENOUGH_MEMORY	0x5550001B	Not enough memory.	The MDR Server / MDR API / MDR Shell is unable to allocate a specified amount of memory.
S_mdr_NOT_INITIALIZED	0x5550001C	MDR server not initialized.	Reserved for future use.
S_mdr_NO_MORE_RECORDER_BLOCKS	0x5550001D	No more space on recorder.	All the space on the storage device is used or the recording size specified is bigger than the available space left on the storage device.
S_mdr_RECORDING_OVERLAP	0x5550001E	Recording overlaps another recording.	The recording specified overlaps another existing recording. This error is reported when creating, resizing or moving a recording (modifying its start block).
S_mdr_RECORDING_OUT_OF_RANGE	0x5550001F	Recording out of range.	The recording number specified is out of the allowed recording number range. In MDR 1.2, this range is from 1 to 248 (included).
S_mdr_RECORDING_NOT_PRESENT	0x55500020	Recording not present.	The specified recording is not present in the recording table.
S_mdr_RECTBL_ACCESS_TIMEOUT	0x55500021	Recording table access time-out	An access (read/write) to the recording table has timed-out. This may indicate a problem with a (several) disk(s) or that the disk(s) has(have) been removed from the FC loop.
S_mdr_RECTBL_INVALID	0x55500022	Recording table is invalid: should be cleared.	The recording table can be set to invalid for several reasons: • The recording table checksum is not consistent with the recording table content, • The recording table contains too many recordings, • The recording table version number does not match the MDR software version number (MDR server version number if the MDR Server accesses the storage device or the MDR API version number if a FC direct access ID used).
S_mdr_RECTBL_NOT_INITIALIZED	0x55500023	Recording table not initialized.	The recording table has not been initialized in the MDR software (MDR Server / MDR API).
S_mdr_REC_NAME_NOT_VALID	0x55500024	MDR recording name not valid.	The recording name specified is invalid. The recording name must not contain any control or special (space, tabulation) characters.
S_mdr_REC_NAME_TOO_LONG	0x55500025	Recording name too long.	The specified recording name is too long. In MDR 1.2, the maximum length of a recording name is 36 characters.
S_mdr_REC_NAME_OUT_OF_RANGE	0x55500026	Recording name number out of range.	The number of the new recording name is out of range. Therefore, it is invalid.
S_mdr_SD_CONTROLLER_ERROR	0x55500027	MDR storage device controller error	An error has occurred during a storage device controller initialization.
S_mdr_SERVER_OCCUPIED	0x55500028	MDR Server is occupied.	Reserved for future use.
S_mdr_SCSI_IOCTL_ERROR	0x55500029	SCSI ioctl command error.	The SCSI tape drive has not been able to execute a SCSI ioctl command.
S_mdr_SCSI_ERROR_REPORT	0x5550002A	Error reporting SCSI error report.	The MDR Server is unable to retrieve the SCSI error report from the SCSI device.
S_mdr_SCSI_TAPE_NOT_PRESENT	0x5550002B	SCSI tape not present.	A tape is expected in the SCSI tape drive but is not present.
S_mdr_SCSI_TAPE_CONFIG_TIMEOUT	0x5550002C	SCSI tape configuration timed-out.	The SCSI tape configuration has timed-out because of an internal SCSI tape drive configuration problem. The configuration has not been done correctly.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

S_mdr_STREAM_DOWN	0x5550002D	TCP/IP stream is down.	The TCP/IP stream between the MDR Server and MDR Client is down. The MDR Server may have shut down the connection due to an internal or communication problem.
S_mdr_SYSTEM_CORRUPT	0x5550002E	MDR system corrupted.	The MDR system is corrupted. This is a critical error. Please contact VMETRO.
S_mdr_TRANSFER_NOT_STARTED	0x5550002F	Transfer not started.	There is no transfer in progress. This indicates that the transfer has not been started or is already finished.
S_mdr_TRANSFER_PREMATURE_END	0x55500030	Transfer finished prematurely.	The MDR has stopped transferring data prematurely (I.e. before having transferring the entire amount of recording data) because an error has occurred. If the transfer was a recording then the recording has been truncated and it contains the recorded data so far (Recording stored and recorded).
S_mdr_TRANSFER_STOPPED	0x55500031	Transfer was stopped.	The transfer phase has been stopped by the MDR Shell command `stop` (or by a call to the MDR API function `StopTransfer`). If the transfer was a recording then the recording has been truncated and it contains the recorded data so far (Recording stored and recorded).
S_mdr_TRANSFER_TIMEOUT	0x55500032	Transfer timed out.	The MDR Server has not been able to transfer (record in this case) the entire recording size because the source: • Has not sent any data before a time greater than the time specified by the `StartInputWaitTime` item (present in the configuration file `mdr.ini`), • Has stopped sending data during a time greater than the time specified by the `InputWaitTime` item (present in the configuration file `mdr.ini`).
S_mdr_UNKNOWN_ERROR	0x55500033	Unknown error.	The error reported is unknown. Usually, this happens when a particular error is encounter. This error is operating system specific. If you are unable to find the signification of this error, please contact VMETRO.
S_mdr_UNSUPPORTED_REQUEST	0x55500034	Unsupported request.	The MDR Server does not support this command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C Upgrading the MDR System Software

This document explains how to upgrade the MDR system (MDR-220 or MDR-250) with new firmware. This upgrade is required in order to update the MDR system with a newer version of the MDR Server program. Also, if the MDR FLASH becomes corrupted, the entire firmware package may be replaced as detailed in the last section.

MDR Firmware (mdrvxst.hex)

The MDR firmware can be found on the VMETRO CD-ROM called ***VMETRO Software Products***.
The `mdr-server` and `mdr-host-sw` directories contain the files needed for this update:

`mdr-server/bin/midas-vxworks/`	contains the file called `mdrvxst.hex` needed for updating the MDR system
`mdr-server/etc/`	contains the configuration files `mdr.ini`, `vxbsp.ini`
`documents/`	contains the documentation, including the ***Release Notes***

Version Upgrade Matrix

The following matrix identifies the software that must be upgraded when migrating from past versions of MDR Software to the current version. At a minimum, the software indicated with a 'Y' must be upgraded.

Currently Installed Version	FC DRIVER		MDR HOST			MDR SERVER (MDR FLASH)		
	Solaris	Windows NT/2000	Solaris	Windows NT/2000	VMFC2x00P Driver	Firmware (mdrvxst.hex)	mdr.ini	vxbsp.ini
mdr 1.1	Y	Y	Y	Y	Y	Y	Y	Y
mdr 1.2.x mdr 1.3.x	Y	Y	Y	Y	Y	Y	Y	N
mdr 1.4 mdr 1.4.1 mdr 1.4.2 mdr-1.4.3	N	Y	Y mdr.ini is same	Y mdr.ini is same	Y	Y	N	N

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10.7.2 Upgrading the MDR firmware using the MIDAS Monitor

Follow the steps below to replace the old MDR firmware with a new version. The upgrade procedure requires a host computer with a serial port (PC or Unix workstation) with a terminal program that supports sending files using the XMODEM protocol.

Make sure you have all the files of the new MDR firmware version available on your host computer. At least the binary file called `mdrvxst.hex` should be present. If the new MDR version requires new configuration files, these should also be available on the host computer. The configuration files are normally readable text files.

Connect the serial line of your host computer to the serial port of the MIDAS board, run the terminal program and power up the MIDAS board. The port settings should be 9600 bauds, no hardware and no software flow control.

Enter the MIDAS Monitor by interrupting the first countdown. Hit <Enter> only once, modify the serial line speed of your terminal program to 38400. Then hit <Enter> a few times and wait for the MIDAS Monitor prompt to appear. The output of the terminal program should be very similar to the following one:

```
------ Start of output --------

Midas i960RP #1 booting ...
i960RP #2      Detected.    Device ID: 0x1960    Vendor ID: 0x8086
i960RP #2      is in Reset.
i960RP #2      Test Passed.
Universe Chip  Detected.    Device ID: 0x0000    Vendor ID: 0x10e3
Universe Status: (Dis, A24, 0, -, -, SysCon)
Universe Chip  Test Passed.
PMC slot 1     Occupied.    Device ID: 0x906d    Vendor ID: 0x10b5
PMC slot 3     Occupied.    Device ID: 0x2100    Vendor ID: 0x1077
PMC slot 5     Occupied.    Device ID: 0x0009    Vendor ID: 0x1011


Boot Test Passed (BDW=00005500)

Connection established...

Hit any key to enter Midas Monitor ...
   1
Starting Midas Monitor.
AutoBaud Detection. Please hit enter a few times....


Midas Monitor 1.02 (Jun  4 1998)

Copyright 1996-98 VMETRO asa


Board Model: M250

Processor : i960RD at 66 MHz; RD step number 01 ; (JF step number 2)

System Information:
  i960RD #    1
  Memory Configuration:
    DRAM :  64 MB
              Address Range: 0xa0000000-0xa4000000
              Monitor Usage:
                       text: 0xa0000000-0xa001e2a4
                       data: 0xa0020000-0xa0021740
                       bss : 0xa0030000-0xa0048d8c
    FLASH:   2 MB
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            Address Range: 0xfe000000-0xfe200000
            Monitor Usage: 0xfe1d0000-0xfe1f9310  Locked.
            MFS Usage    : 0xfe1c0000-0xfe1d0000
  Serial Port Baud Rate:       38400
#
------ End of output --------
```

Update the configuration files

Check if the new configuration files `vxbsp.ini` and `mdr.ini` (`mdr_server.ini` and/or `mdr_host.ini` according to your configuration) are different from than the configuration files present on your MDR system. If so then transfer the new configuration files (`vxbsp.ini` and/or `mdr.ini`) one by one as shown in the following method below:

a) First, the files which need to be updated have to be deleted as follow:

`# rm vxbsp.ini` (only if `vxbsp.ini` needs to be updated)
`# rm mdr.ini` (only if `mdr.ini` needs to be updated)

b) The configuration files should be transferred one by one with the `mgf` command. Both the file name and the file size need to be specified as arguments (because the XMODEM protocol does not support communication of these parameters) as follows:

`# mgf vxbsp.ini 225` (only if `vxbsp.ini` needs to be updated)
`# mgf mdr.ini 766` (only if `mdr.ini` needs to be updated)

Note: The file sizes used above are given as examples and should not be used. The real sizes of the configuration files present on the ***VMETRO Software Products*** CD-ROM may differ. Please find out the size of these files before trying to download them.

The `mmon.ini` file supplied by VMETRO is a generic file and must **NOT** be uploaded in any case. The items contained in this file MUST be updated using the **`iset`** command.

Erase the Flash Memory of the MIDAS board by using the command `ef` as follows:

```
# ef fe000000 1c0000
```

Start the downloader in the MIDAS Monitor by using the `dos` command as follows:

```
# dos fe000000
```

Send the `mdrvxst.hex` file across the serial line from your terminal program using the XMODEM protocol.

If your system has more than one MDR board, steps above should be repeated.

Modify the serial line speed of your terminal program 9600 bauds and reset the MDR board. The MDR system will reboot, the MDR Server will start automatically and, after an initialization phase, will wait for a client to connect.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Restoring Corrupt FLASH

If the MDR FLASH is corrupted, the following procedure may be used to recover the MDR board:

Delete the following configuration files from the MDR board:

```
# rm mdr.ini
# rm vxbsp.ini
```

Do **not** delete the `mmon.ini` file.

Step #1:
Reboot the board and press <Enter> to enter the Midas Monitor.
The output from your board will look something like this:

```
Midas i960RP #1 booting ...
UART           Test Passed.
Test Type Boot Test (   1)
Testing Memory   /
DRAM           Test Passed.
i960RP #2      Detected.    Device ID: 0x1960    Vendor ID: 0x8086
i960RP #2      is in Reset.
i960RP #2      Test Passed.
Universe Chip  Detected.    Device ID: 0x0000    Vendor ID: 0x10e3
Universe Status: (Dis, A24, 0, -, -, -)
Universe Chip  Test Passed.
PMC slot 1     Occupied.    Device ID: 0x2100    Vendor ID: 0x1077

Boot Test Passed (BDW=00007c00)
Connection established...
Hit any key to enter Midas Monitor ...
   3
Starting Midas Monitor.
AutoBaud Detection. Please hit enter a few times....

Midas Monitor 1.02 (Jun  4 1998)
Copyright 1996-98 VMETRO asa
Board Model: M220SR
Processor : i960RP at 33 MHz; RP step number 01; (JF step number 02)
System Information:
  i960RP #    1
  Memory Configuration:
    DRAM :  64 MB
             Address Range: 0xa0000000-0xa4000000
             Monitor Usage:
                      text: 0xa0000000-0xa001e2a4
                      data: 0xa0020000-0xa0021740
                      bss : 0xa0030000-0xa0048d8c
    FLASH:   1 MB
             Address Range: 0xfe000000-0xfe100000
             Monitor Usage: 0xfe0d0000-0xfe0f9310  Locked.
             MFS Usage    : 0xfe0c0000-0xfe0d0000
   Serial Port Baud Rate:       9600
```

Step #2:
Type `sc` at the # prompt.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You will see the following prompt:
"Are you absolutely sure you want to clear the MFS Flash block?"
Press `y`.

You will then see:
```
Please wait while clearing the MFS Flash block
```
Type `quit`.
The board will then reboot.

Step #3:
At the following message, hit <Return> twice:
```
AutoBaud Detection. Please hit enter a few times....
```

You will then see:
```
Do you want to generate mmon.ini manually? (y/n) :
```
Type `n`.

Respond `y` when asked you if you want to recover `mmon.ini` from EPROM.

Reboot the board.

Upgrade the MDR firmware as outlined above.

Upgrading the MDR host software

When upgrading the MDR firmware, it is required to also upgrade the MDR host software for all the platforms that are used. When connecting, the MDR software performs a version consistency check. If the MDR firmware is updated but not the MDR host software then the MDR host software (MDR Shell or user's program using the MDR API) will not connect to the MDR Server and exit with an error.

Follow the steps below to replace the old MDR host software with the new version.

1. Locate the directory where the old MDR host software has been installed. For instance, `/usr/mdr-host-sw/bin/sun-solaris` if the selected platform is `sun-solaris` and the MDR root directory is `/usr`.
2. On the ***VMETRO Software Products*** CD-ROM, locate the MDR host software binary file. This file can be found in `mdr-host-sw/bin/<platform>` and is called `mdrshell` (or `mdrshell.exe` for the `x86-win32` platform).
3. Replace the old MDR host software binary file on your host computer with the new file found on the ***VMETRO Software Products*** CD-ROM.
4. If required, replace the old MDR host software library file(s) on your host computer with the new file(s) found on the ***VMETRO Software Products*** CD-ROM. The new library file(s) can be found in `mdr-host-sw/lib/<platform>` on the ***VMETRO Software Products*** CD-ROM.
5. Set up the environment variables according to your platform:
   - `MDR_BASE` should be set to your local MDR base directory for all the platforms
   - `LD_LIBRARY_PATH` should contain the path to the `<mdr-base-dir>/lib/sun-solaris/.libs` directory for `sun-solaris`.
6. Execute the new MDR host software and try to connect to the MDR.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D Supported PMCs Documentation

This appendix describes the PMCs that are supported by the MDR system.

Fibre Channel Storage

The permanent storage medium used by MDR is based on commercial hard disks using the FC-AL (Fibre Channel Arbitrated Loop) interface. Fibre Channel is a serial link, using either optical cables or differential ECL twinax cables, operating at 1.062 Gbit/sec. The peak effective transfer rate of a Fibre Channel loop is approximately 100 MiB/sec. MDR gets its disk recording performance from using an array of Fibre Channel disks such as **RAID** (Redundant Arrays of Independent Disks) or **JBOD** (Just a Bunch Of Disks) units.

VMFC-2100 Fibre Channel PMC

The VMFC-2100 is a Fibre Channel PMC module specifically designed to maximize throughput while minimizing transfer latency and host processor overhead in embedded real-time applications. In addition to mainstream Fibre Channel applications such as disk storage, this interface is particularly well suited to applications where low latency communications and high-sustained throughput are essential. This includes areas such as multi-processor communications (telecommunications & radar signal processor), sensor I/O (radar, sonar & image processing), and ultra-high performance computer networks (digital broadcasting & subsystems data links). For further information on the VMFC-2100 PMC module, see the *VMFC-2100 Fibre Channel PMC Module User's Guide.*

Current Models

VMFC-2100-DC	Fibre Channel PMC module with dual copper HSSDC interface and integrated hub.
VMFC-2100P-DC	Fibre Channel PCI card with dual copper HSSDC interface and integrated hub.
VMFC-2100-CF	Fibre Channel PMC module with single copper and single fibre-optic interface and integrated hub.
VMFC-2100P-CF	Fibre Channel PCI card with single copper and single fibre-optic interface and integrated hub.
VMFC-2100-DF	Fibre Channel PMC module with dual fibre-optic interface and integrated hub.
VMFC-2100P-DF	Fibre Channel PCI card with dual fibre-optic interface and integrated hub.

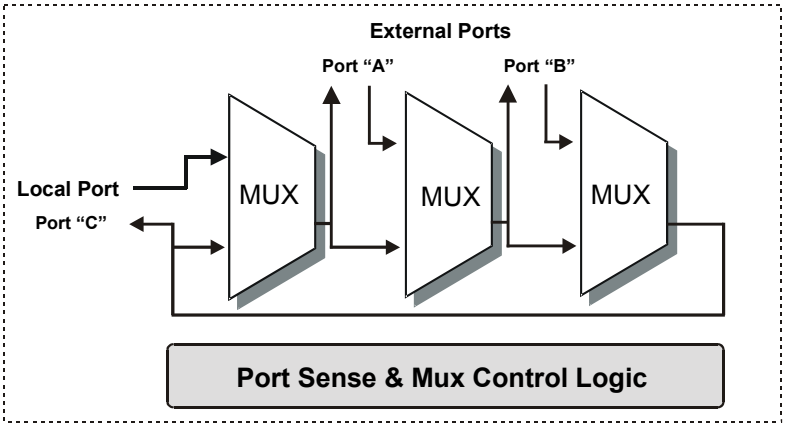
Single Arbitrated Loop (RAID/JBOD connection)

The single arbitrated loop connection from the VMFC-2100 to the RAID/JBOD, is the simplest point-to-point communications link possible with the VMFC-2X00.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OmniPort Integrated Hub Interconnection

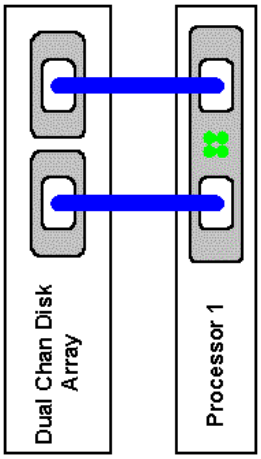
The integrated hub capabilities of the Patent pending OmniPort Fibre Channel I/O port allows for simple cascading of multiple Fibre Channel devices. Standard duplex HSSDC copper and SC Duplex fibre optic cabling is supported. Three modes of operation include integrated hub, redundant point to point and redundant arbitrated loop (JBOD connection).



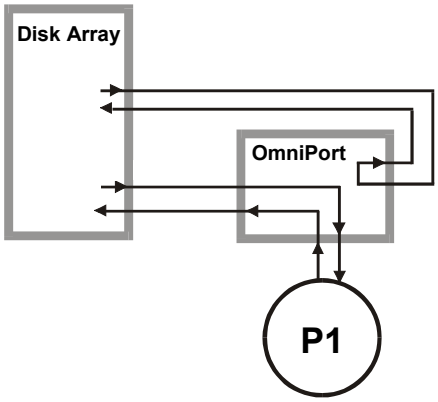
Intergrated Hub block diagram

Redundant Arbitrated Loop (RAID/JBOD connection)

With redundant connections from the VMFC-2X00 to the RAID/JBOD, the OmniPort automatically switches to the alternate loop when the current loop fails. The alternate loop is looped back to the disk array. The OmniPort will park on the last known good loop and only switch upon failure detection. Redundant connections improve the MTBF of the system.



Redundant Arbitrated Loop connection



Redundant Arbitrated Loop functional diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VMFC-2100 Specifications

Fibre Channel	FC-AL rev 4.5
	FC-PH rev 4.3
	SCSI_FCP rev 12
PMC, PMC Host Bus	PCI Local Bus revision 2.1
	PMC IEEE P1386.1
	CMC IEEE P1386
Operating Temperature	0 to 70 °C Greater than 50 °C operation requires adequate airflow (≥100 LFPM)
Storage Temperature	-40 to +85 °C
Operating Humidity	5% to 95% non-condensing
Storage Humidity	5% to 95% non-condensing
Power	+5 Vdc @ 1.3 amps max Dual Copper
	+5 Vdc @ 1.7 amps max Dual Fibre
Weight	4 oz.
Warranty	1 year limited warranty

Product design and specifications are subject to change without notice.

Front Panel Indicators

There are four LED indicators on the front panel of the Host Adapter assigned as follows:

AUT The OmniPort is configured for automatic operation. (Default)
LCL The Host Adapter Fibre Channel controller is active.
P1 Valid Fibre Channel signaling is detected on external port 1.
P2 Valid Fibre Channel signaling is detected on external port 2.

Fibre Channel Cables

The following cables can be ordered directly from VMETRO.

Part Number	Description
FCC-HD-03	3 meter Fibre Channel HSSDC to 9 Pin DB9 Copper Cable
FCC-HD-05	5 meter Fibre Channel HSSDC to DB9 Copper Cable
FCC-HD-10	10 meter Fibre Channel HSSDC to DB9 Copper Cable
FCC-HD-20	20 meter Fibre Channel HSSDC to DB9 Copper Cable
FCC-HAD	0.5 meter (1.6 ft) Fibre Channel HSSDC to DB9 Adapter (for use from PMC to chassis-mount DB9 female)
FCC-DD-03	3 meter 9 Pin Plug to 9 Pin D Plug FC Copper Cable
FCC-DD-05	5 meter 9 Pin Plug to 9 Pin D Plug FC Copper Cable
FCC-DD-10	10 meter 9 Pin Plug to 9 Pin D Plug FC Copper Cable
FCC-DD-20	20 meter 9 Pin Plug to 9 Pin D Plug FC Copper Cable

For other cable lengths, contact VMETRO.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FPDP - Digital I/O

An MDR system normally interfaces to its data source with a digital I/O interface based on the industry standard FPDP (Front Panel Data Port) specification (Proprietary ports may also be accommodated; please consult factory). The FPDP interface is a 32-bit synchronous input/output parallel interface which is specified to operate at clock rates up to 25 MHz (TTL). The FPDP interface is used by a variety of third party vendors with products such as A/D converters, D/A converters and DSP boards.

DPIO FPDP PMC

The DPIO (Digital Parallel I/O) PMC (PCI Mezzanine Card) module is designed for high-speed data acquisition and generation. It combines a Digital Parallel Input or Output on one end, a 32-bit PCI bus master/slave interface with a DMA controller on the other end, with a large FIFO in between. For more detailed information about the DPIO PMC module, see the *DPIO PMC Module User's Manual*

DPIO Features

- 32-bit input/output FIFOs,
- Synchronous port with FPDP interface,
- Linked-list DMA controller on PCI,
- Personality module for proprietary and user-defined I/O interfaces,
- PLD allows customization, data packing,
- Optional byte swapping,
- 16-bit, 10-bit, 8-bit or 4-bit data packing,
- Up to 108 MiB/sec PCI data rate (sustained).

Current Models

DPIO-FI & DPIO-FO

FPDP Compliant input and output models respectively for TTL level signaling.

- 32-bit parallel I/O TTL up to 25 MHz
- 16-bit parallel I/O TTL up to 40 MHz

DPIO-EI & DPIO-EO

Input and output models for high speed differential signaling using PECL technology.

- 16-bit parallel I/O up to 50 MHz
- 10-bit parallel I/O up to 70 MHz

DPIO-LI & DPIO-LO

Input and output models for differential signaling using LVDS signaling.

- 32-bit parallel I/O LVDS up to 25 MHz
- 16-bit parallel I/O LVDS up to 46.5 MHz

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DPIO-DI & DPIO-DO
Input and output models for differential signaling using RS422 signaling.

- 32-bit parallel I/O RS422 up to 10 MHz

Packing Options

The DPIO can be set up to use different packing options. This is done by reloading a "Packing PLD" on the DPIO. VMETRO provides a PLD Reprogramming Kit (part no. PLD-KIT-DPIO) for this purpose. The kit contains everything that is needed: a PC parallel port cable, JEDEC files for all packing options, and software.

Input/Output port	Packing	PCI
32 bit (D31:D00)	No packing	32 bit
16 bit (D15:D00)	2 to 1 packing	32 bit
10 bit (D09:D00)	3 to 1 packing	30 bit (databits 31 or 30 not in use)
8 bit (D07:D00)	4 to 1 packing	32 bit
4 bit (D03:D00)	8 to 1 packing	32 bit

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DPIO Specifications

Max. input data rate	25 MHz at 32-bits (-FI/FO versions)
	40 MHz at 16-bits with 2:1 packing (-FI/FO versions with PECL strobe)
	50 MHz at 16-bits with 2:1 packing (-EI/EO versions only)
	75 MHz at 10-bits with 3:1 packing (-EI/EO versions only)
	75 MHz at 8-bits with 4:1 packing (-EI/EO versions only)
	49 MHz at 32-bits (-LI/LO versions)
Max. PCI transfer rate	108 MiB/s
FIFO size	8Kx32 or 32Kx32
Cable lengths	1.0m(max), 0.3m, 0.1m (TTL) 10m(max), 5m, 2m (RS422/LVDS)
Operating Temperature	0 to 70 °C Greater than 50°C operation requires adequate airflow (≥ LFPM)
Storage Temperature	-40 to +85 °C
Operating Humidity	5% to 95% non-condensing
Storage Humidity	5% to 95% non-condensing
Power consumption	Typical 1.0A / max 1.5A at +5V
Weight	100 grams

DPIO-FI/FO - FPDP

The most common interface for the DPIO is the FPDP (Front Panel Data Port). The FPDP bus is intended to provide data transfer between two or more VMEbus boards at up to 160 MiB/s with the lowest possible latency, without compromising existing VMEbus and other connections on the chassis P1 and P2 connectors. FPDP is connected by means of an 80-conductor ribbon cable connector at the front panel of the VMEbus board. The wiring topology is in the form of a bus. Multiple FPDP buses may coexist in a single VMEbus enclosure.

Details about FPDP can be found in the FPDP Specification at VITA's web site: http://www.vita.com.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Connector Pin Assignments (non-inverted)

PIN	ROW A	ROW B	ROW C	ROW D
1	GND	STROB	GND	GND
2	GND	GND	/NRDY	GND
3	/DIR	GND	RESERVED (RES1)	GND
4	/SUSPEND	GND	GND	GND
5	PIO2	GND	PIO1	GND
6	RESERVED (RES2)	GND	RESERVED (RES3)	GND
7	PSTROB	GND	/PSTROB	GND
8	/SYNC	GND	/DVALID	GND
9	D31	D30	GND	D29
10	D28	GND	D27	D26
11	GND	D25	D24	GND
12	D23	D22	GND	D21
13	D20	GND	D19	D18
14	GND	D17	D16	GND
15	D15	D14	GND	D13
16	D12	GND	D11	D10
17	GND	D09	D08	GND
18	D07	D06	GND	D05
19	D04	GND	D03	D02
20	GND	D01	D00	GND

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Signal Termination

Signal	DPIO-FI	DPIO-FO	DPIO-EI	DPIO-EO	DPIO-DI/LI	DPIO-DO/LO
SUSPEND*	330Ω pull down 220Ω pull up	330Ω pull down 220Ω pull up	390Ω to GND on both + and - wires	110Ω across + and – wires		110Ω across + and – wires
DIR*	330Ω pull down 220Ω pull up	330Ω pull down 220Ω pull up	NA	NA	NA	
NRDY*	330Ω pull down 220Ω pull up	330Ω pull down 220Ω pull up	NA	NA		110Ω across + and – wires
STROB	330Ω pull down 220Ω pull up	330Ω pull down 220Ω pull up	110Ω across + and – wires	390Ω to GND on both + and - wires	110Ω across + and – wires	
PSTROB PSTROB*	110Ω across + and – wires	390Ω to GND on both + and - wires	NA	NA	NA	
DATA		27Ω series	110Ω across + and – wires	390Ω to GND on both + and - wires	110Ω across + and – wires	
DVALID		27Ω series	110Ω across + and – wires	390Ω to GND on both + and - wires	110Ω across + and – wires	
SYNC		27Ω series	110Ω across + and – wires	390Ω to GND on both + and - wires	110Ω across + and – wires	
PIO2:1	27Ω series¤	27Ω series	110Ω across + and – wires	390Ω to GND on both + and - wires	110Ω across + and – wires	110Ω across + and – wires
RES3	27Ω series¤	27Ω series	NA	NA	110Ω across + and – wires	
RES2:1	27Ω series¤	27Ω series	NA	NA	NA	NA

¤ *when connected*

FPDP Options

The FPDP standard defines a number of signals, which may be used differently in different modes of operation. This section describes their usage in the MDR.

Clocking

Input module (DPIO-FI)

Two separate clocks are provided on the FPDP. STROB is a single-ended clock conforming to TTL levels, while PSTROB and /PSTROB is a differential Positive Emitter-Coupled Logic version of the same clock. Either of these signals may be used on a FPDP input module. However, the PSTROB, /PSTROB signal pair is preferable, especially at higher clocking frequencies and when driving longer lines, since the noise margin is improved over the TTL clock. A control register bit on the DPIO selects between the two clocks. This choice between these two sorts of signals will be available from the version 1.2 of the MDR.

Output module (DPIO-FO)

A replaceable oscillator is used to generate clocks when operating as data source. The oscillator frequency can be used directly or be divided by 2, 4, or 8 by means of control register bits. This can be setup with the item called `OutputFreqDivisor` in the `DpioInfo` section of the `mdr.ini` file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SUSPEND

/SUSPEND is generated by the receiver to inform the data source of a pending FIFO overflow condition. The data source is allowed as many as 16 cycles before suspending the transfer. Since /SUSPEND is asynchronous to STROB, the data source should synchronize it before sampling its state; this avoids stability problems.
This signal is active low :

- Driven to logical "0" to activate SUSPEND,
- Driven to logical "1" to deactivate SUSPEND.

The DPIO of the MDR may activate the Suspend signal to indicate that the DPIO FIFO is almost full so that the data source should stop sending data until the Suspend signal is deactivated again. The user may enable or disable the use of the Suspend signal by setting the `SuspendSignal` item in the `mdr.ini` file.

To disable the Suspend signal, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo SuspendSignal Disable mdr.ini
```

To enable the Suspend signal, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo SuspendSignal Enable mdr.ini
```

PIO Bits

The DPIO input modules use the PIO signals for synchronization with the data source when starting/stopping the recording. Generally, either PIO1 or PIO2 signal can be used. The PIO signal is configured as an output signal from the DPIO input module to signal the data source that the MDR is ready to start recording/receiving data. The data source is responsible for not sending data until the PIO signal has been asserted (logical level "1"). When the recording has finished, the MDR will de-assert PIO (logical level "1" to "0") to signal that it is so.
In future versions, the MDR will support functions in the API to allow the user to control the PIO bits.

NRDY (Not Ready)

The DPIO input module uses NRDY to signal to the data source that a recording has ended. The NRDY signal works almost the same way as the PIO bits. The difference is that the NRDY signal is deactivated (DPIO ready to receive) a few micro-seconds (approximately 3μs) before the PIO signal is activated. The reason for this apparent redundancy is that the NRDY signal, which is the real data ready signal, is frequently not available for the user to be used as a system start signal. It is buried at a low hardware level. However, the PIO, which is a user-controlled signal, is more likely to be available to the user.
Please note that the DPIO input hardware has no way of activating NRDY right after the last word of the last recording block has been received. The NRDY bit may only be controlled by software, and the software has no way of knowing the exact number of words having been received at any time by the DPIO input module. The DMA controller

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

is used to notify the software when a specified amount of data has been transferred to the MIDAS DRAM.

SYNC

The SYNC* signal can be used to synchronize the DPIO input module installed on the MDR system with the transmitter. It is used in 'single frame data' mode.
The SYNC* signal is asserted prior to the transmission of data by the data transmitter, not at the same time as VALID*. This is to synchronize the receiver to the source. SYNC* is ignored by the receiver after the first assertion.

To disable the use of the SYNC* signal, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo WaitForSync Disable mdr.ini
```

To enable the use of the SYNC* signal, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo WaitForSync Enable mdr.ini
```

FPDP Clocks

Two separate clocks are provided on the FPDP. STROB is a single-ended clock confirming to TTL levels, while PSTROB and /PSTROB is a differential positive Emitter-Coupled Logic (ECL) version of the same clock. Either of these signals may be used. However, the PSTROB /PSTROB signal pair is preferable, especially at high clocking frequencies and when driving longer lines, since the noise margin is improved over the TLL clock.

To disable the use of the PSTROB /PSTROB signal pair, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo PeclStrobe Disable mdr.ini
```

To enable the use of the PSTROB /PSTROB signal pair, type the following command at the MIDAS Monitor prompt:

```
# iset DpioInfo PeclStrobe Enable mdr.ini
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DPIO Cables

Part Number	Description
CBL-FPDP-10	FPDP 80 pin Ribbon Cable w/connectors, 10cm
CBL-FPDP-30	FPDP 80 pin Ribbon Cable w/connectors, 30cm
CBL-FPDP-100	FPDP 80 pin Ribbon Cable w/connectors, 100cm
CBL-FPDP-10I	Inverted FPDP Ribbon Cable w/connector (for MCS RIN-T/ROUT-T), 10cm
CBL-FPDP-30i	Inverted FPDP Ribbon Cable w/connector (for MCS RIN-T/ROUT-T), 30cm
CBL-FPDP-100i	Inverted FPDP Ribbon Cable w/connector (for MCS RIN-T/ROUT-T), 100cm
CBL-RS422-2M	RS422 Twisted Pair 80 pin Ribbon Cable w/connectors, 2m
CBL-RS422-5M	RS422 Twisted Pair 80 pin Ribbon Cable w/connectors, 5m
CBL-RS422-10M	RS422 Twisted Pair 80 pin Ribbon Cable w/connectors, 10m
CBL-PECL-30	PECL Twisted Pair 50 pin Ribbon Cable w/connectors, 30cm
CBL-PECL-60	PECL Twisted Pair 50 pin Ribbon Cable w/connectors, 60cm
CBL-PECL-90	PECL Twisted Pair 50 pin Ribbon Cable w/connectors, 90cm

Proprietary Interfaces - PECL/LVDS/RS422

The DPIO is available also in models with proprietary front panel interfaces. These are:

DPIO-EI/EO High speed differential signaling - PECL
DPIO-LI/LO High speed differential signaling - LVDS
DPIO-DI/DO Differential signaling - RS422

Although the interfaces have proprietary pin-outs defined by VMETRO, they are based on the protocol and signal names of FPDP. Detailed information on these interfaces can be found in the *DPIO PMC Module User's Manual*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Ethernet Host Interface

The MDR module utilizes the Ethernet PMC to provide networking access to hosts like a PC or workstation.

DE520 Ethernet PMC

The DE520 is an Ethernet Adapter as a PCI Mezzanine Card (PMC) with a direct interface to the 32-bit PCI local bus. This dual-speed module uses a single connector for either a 10 megabits per second (10 Mb/s) or a 100 megabits per second (100 Mb/s) IEEE 802.3 Ethernet network connection. The module automatically senses and switches to the correct speed.

Current Model

DE520 Ethernet PMC Module, 10 or 100 Mb/s from Compaq.

Features

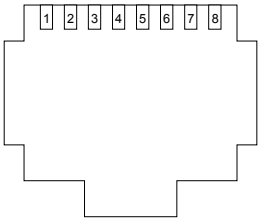
The DE520 Ethernet PMC Module has the following features:

- High performance: full-speed, 32-bit DMA bus transfers at 132 MiB/s.
- Single-shielded or unshielded RJ-45 connection.
- Automatic sensing of speed and change to that speed.
- Powerful parallel cut-through architecture and other patented features to maximize data throughput and minimize CPU use.
- 100BASE-TX (100 Mb/s) with UTP Category 5 cabling or 10BASE-T (10 Mb/s) with UTP Category 3, 4, 5 cabling.
- Configurable by software to operate in full-duplex mode, increasing the aggregate bandwidth up to 20 Mb/s and 200 Mb/s.
- Compliance with PCI revision 2.0.
- Compliance with FCC Class A and CISPR-22 Class A.

Connector Pinout (RJ-45)

Pin Number	Signal Name
1	Transmit +
2	Transmit –
3	Receive +
4	NC (no connection)
5	NC (no connection)
6	Receive –
7	NC (no connection)
8	NC (no connection)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



RJ-45 Connector (Front View)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCSI PMC Backup

The MDR uses the SCSI PMC module to provide connectivity to SCSI tape stations for backup purposes.

SC-PMC/8X5 SCSI PMC Module

The SC-PMC/8X5 (X is 2 or 7) is a SCSI Adapter as a PCI Mezzanine Card (PMC) with a direct interface to the 32-bit PCI local bus. The SC PMC/8X5 supports both SCSI-I and SCSI-II. A narrow or wide single-ended interface is provided with both asynchronous or synchronous operation.

Current Model

SC-PMC/875 SCSI PMC Module from Concurrent Technologies.

Features used on the MDR system

The SC PMC/875 SCSI PMC Module has the following features:

- High performance: 32-bit DMA bus transfers at 110 MiB/s.
- SCSI-II
- Ultra Wide single-ended interface
- Asynchronous or synchronous operation
- Active termination and signal negation
- Utilizing NCR's 53C875 SCSI I/O
- Synchronous SCSI data rates up to 40MiB/sec

Supported tape drives

The MDR has been tested successfully with the following tape drives:

- Seagate Sidewinder 50 AIT-1/2
- Quantum DLT 7000.

SCSI Tape Format

The MDR system can be used with a SCSI PMC module connected to a Wide SCSI tape drive. Using the MDR Shell (or API) functions, it is possible to copy data from the storage device to the tape drive and vice versa.

A recording can be saved on one or several tapes according to the tape capacity and the recording size.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data Format

The data format on tape is the following:

- Tape header which is a MASTER_TAPE_HEADER structure:

```
typedef struct {
  UINT32    TapeNumber; /* Number of the tape (1 to n)   */
  UINT32    DataLength; /* Data length on tape in KiB */
} TAPE_HEADER;

typedef struct {
TAPE_HEADER TapeHeader; /* Tape information      */
char        Name[RECORDING_NAME_MAXLENGTH];
RECORDING   Recording;  /* Recording information */
} MASTER_TAPE_HEADER;
```

  Where `RECORDING_NAME_MAXLENGTH` and `RECORDING` are defined in the `MdrDef.h` file.

- Recording data saved as raw and compressed data. Data is send to the drive uncompressed, compressed internally inside the drive (if possible) then saved to the tape.

Reading a recording to host

Reading a recording from tape(s) to host can be done using SCSI commands to:

1. Open and configure the tape drive (the current block size used is 1kbytes),
2. Read the Tape Header structure from tape,
3. Read raw data from tape(s). Data size on each tape is given by the `TAPE_HEADER` structure) and the number of tapes is computed by subtracting the data length for each tape,
4. Close the tape drive.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Front Panel Connector Pinout (68-way SCSI connector)

Pin No.	Signal Name	Pin No.	Signal Name
1	0V	35	SCSID12*
2	0V	36	SCSID13*
3	0V	37	SCSID14*
4	0V	38	SCSID15*
5	0V	39	SCSIP1*
6	0V	40	SCSID0*
7	0V	41	SCSID1*
8	0V	42	SCSID2*
9	0V	43	SCSID3*
10	0V	44	SCSID4*
11	0V	45	SCSID5*
12	0V	46	SCSID6*
13	0V	47	SCSID7*
14	0V	48	SCSIP0*
15	0V	49	0V
16	0V	50	0V
17		51	TERMPWR
18		52	TERMPWR
19		53	
20	0V	54	0V
21	0V	55	SCSIATN*
22	0V	56	0V
23	0V	57	SCSIBSY*
24	0V	58	SCSIACK*
25	0V	59	SCSIRST*
26	0V	60	SCSIMSG*
27	0V	61	SCSISEL*
28	0V	62	SCSICD*
29	0V	63	SCSIREQ*
30	0V	64	SCSIIO*
31	0V	65	SCSID8*
32	0V	66	SCSID9*
33	0V	67	SCSID10*
34	0V	68	SCSID11*

** Denotes signal active low*

SC PMC/875 Specifications

SCSI	SCSI-I and SCSI-II Narrow (8-bit), wide (16-bit) or ultra wide (32-bit) single-ended interface
PMC, PMC Host Bus	PCI Local Bus revision 2.1 PMC IEEE P1386.1 CMC IEEE P1386
Operating Temperature	0 to 55 °C with 200LFM air flow
Storage Temperature	-40 to +70 °C
Operating Humidity	10% to 90% non-condensing
Storage Humidity	10% to 90% non-condensing
Power	+5 Vdc @ 0.2A (excludes SCSI bus requirements)
Dimensions	Height: 13.5 mm Depth: 149 mm Width: 74 mm Weight: 90g max.

Product design and specifications are subject to change without notice.

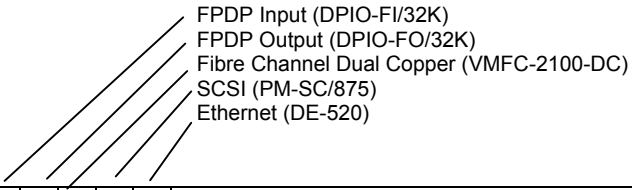
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E MDR Standard Configurations

MDR-220-xx **x = PMC Slot 1 or 2**
MDR-250-xxxxx **x = PMC Slot 1, 2, 3, 4 or 5**

x = 0	PMC Slot Not Populated	
x = 1	FPDP Input	(DPIO-FI/32K)
x = 2	FPDP Input / 16 bit Packing	(DPIO-FI/32K/16bit)
x = 3	FPDP Output	(DPIO-FO/32K)
x = 4	FPDP Output / 16 bit Packing	(DPIO-FO/32K/16bit)
x = 5	Dual Copper FC	(VMFC-2100-DC)
x = 6	Copper & Fibre FC	(VMFC-2100-CF)
x = F	Dual Fibber FC	(VMFC-2100-DF)
x = 7	Serial FPDP Input / SWL	(FHG2-PM4MWR02-00) (Not supported yet)
x = 8	Serial FPDP Output / SWL	(FHG2-PM4MWT02-00) (Not supported yet)
x = 9	Serial FPDP Bidir. / SWL	(FHG2-PM4MWB02-00) (Not supported yet)
x = E	Fast Ethernet	(DE520)
x = S	Ultra Wide SCSI	(SC-PMC/875)
x = C	Custom configuration	

Standard Configurations:



MDR-220-15	✓		✓			MDR-220 Data Recorder Module. FPDP Input, Dual Copper FC
MDR-220-35		✓	✓			MDR-220 Data Playback Module. FPDP Output, Dual Copper FC
MDR-250-1050E	✓		✓		✓	MDR-250 Data Recorder Module. FPDP Input, Dual Copper FC, Ethernet
MDR-250-105S0	✓		✓	✓		MDR-250 Data Recorder Module. FPDP Input, Dual Copper FC, SCSI
MDR-250-105SE	✓		✓	✓	✓	MDR-250 Data Recorder Module. FPDP Input, Dual Copper FC, SCSI and Ethernet
MDR-250-0350E		✓	✓		✓	MDR-250 Data Playback Module. FPDP Output, Dual Copper FC, Ethernet
MDR-250-035S0		✓	✓	✓		MDR-250 Data Playback Module. FPDP Output, Dual Copper FC, SCSI
MDR-250-035SE		✓	✓	✓	✓	MDR-250 Data Playback Module. FPDP Output, Dual Copper FC, SCSI and Ethernet
MDR-250-13500	✓	✓	✓			MDR-250 Data Recorder & Playback Module. FPDP Input & Output, Dual Copper FC
MDR-250-1350E	✓	✓	✓		✓	MDR-250 Data Recorder & Playback Module. FPDP Input & Output, Dual Copper FC, Ethernet
MDR-250-135S0	✓	✓	✓	✓		MDR-250 Data Recorder & Playback Module. FPDP Input & Output, Dual Copper FC, SCSI
MDR-250-135SE	✓	✓	✓	✓	✓	MDR-250 Data Recorder & Playback Module. FPDP Input & Output, Dual Copper FC, SCSI and Ethernet

Other configurations than those listed above are possible. Please consult factory for assistance in creating the model that suits your requirements.

Restrictions:

Ethernet	Slot 5
Fast Wide SCSI	Slot 3 or 4

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

F Miscellaneous Functionalities

MDR Status Block mapped to VMEbus

A shared memory region is mapped by the MDR on the VMEbus and can be used during the recording phase.
This shared region contains information on:

- The MDR Server itself: signature, version numbers and MDR model name,
- Where are mapped (on the VMEbus) the Progress Indicator structure and the Control Data structure,
- The state of the Progress Indicator (Enabled or Disabled),
- The recording phase (progress indication).

The following structure can be read from the VMEbus:

```
typedef struct SHARED_MEMORY_HEADER {
  UINT32    MdrSignature;
  UINT16    MajorVersion;
  UINT16    MinorVersion;
  UINT8     MdrModelName[MDR_MODEL_NAME_MAXSIZE];
  UINT32    ProgressIndicatorEnabled;
  UINT32    MonitorDataStructureOffset;
  UINT32    ControlDataStructureOffset;
  UINT32    SharedMemoryVMEBaseAddress;
} SHARED_MEMORY_HEADER;
```

Currently,

- `MdrSignature` is set to `0xBEC0BDEB`,
- `MajorVersion` and `MinorVersion` are set to `1` and `2`,
- `MdrModelName` is your MDR model name (defined in the `mdr.ini` file) and `MDR_MODEL_NAME_MAXSIZE` is defined in the `MdrDefs.h` file,
- `ProgressIndicatorEnabled` is a flag indicating, if `TRUE`, that the progress indicator structure (mapped to the VMEbus) contains valid data and is updated by the MDR Server. If `FALSE`, the progress indicator structure contains invalid data and should be ignored,
- `MonitorDataStructureOffset` pointed to a structure which is defined as follows:

```
typedef struct MDR_PROGRESS_INDICATION {
  UINT32    Running;     /* TRUE if recording/playback is running */
  UINT32    TotalBlocks; /* Total number of recorder blocks in
            current running recording/playback */
  UINT32    BlockNo;     /* recorder block being processed */
} MDR_PROGRESS_INDICATION;
```

- The `ControlDataStructureOffset` parameter is not currently used.

The `SharedMemoryDataVmeBaseAdrs` structure is mapped on the VMEbus by the MDR during the initialisation phase and can be read (only) at any time while the MDR Server program is running.
In the `mdr.ini` configuration file, the item called `SharedMemoryDataVmeBaseAdrs` is used to specify the VME base address of the `SHARED_MEMORY_HEADER` structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MDR Server boot status

During each reboot, an automatic Built-In Test (BIT) is carried out to ensure that the MDR board and the MDR Server are working correctly.

Built-In Test Description

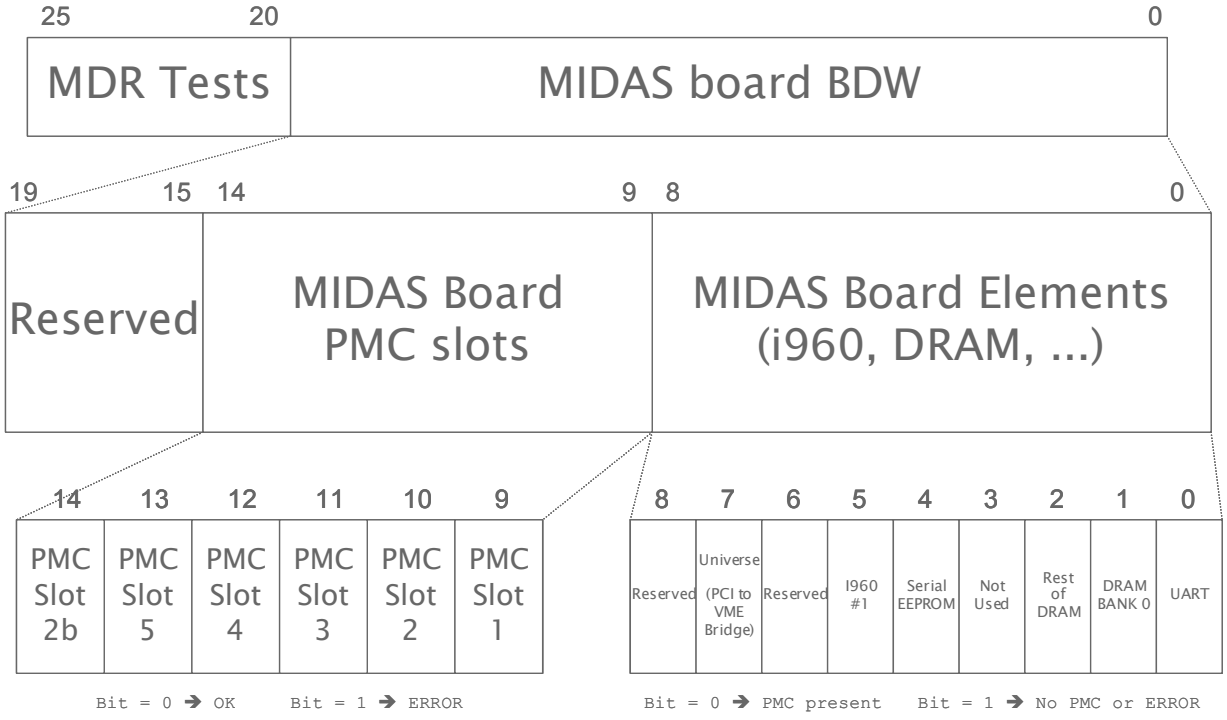
The MDR BIT covers the following components of the MDR board:

- MIDAS board Test carried out by the MIDAS Monitor (at boot up)
- SCSI PMC Test carried out by the BSP and MDR Server (initialization)
- DPIO PMCs Test carried out by the MDR Server (initialization)
- VMFC-2100 PMC Test carried out by the MDR Server (initialization)
- Ethernet PMC Test carried out by the MDR Server (initialization)

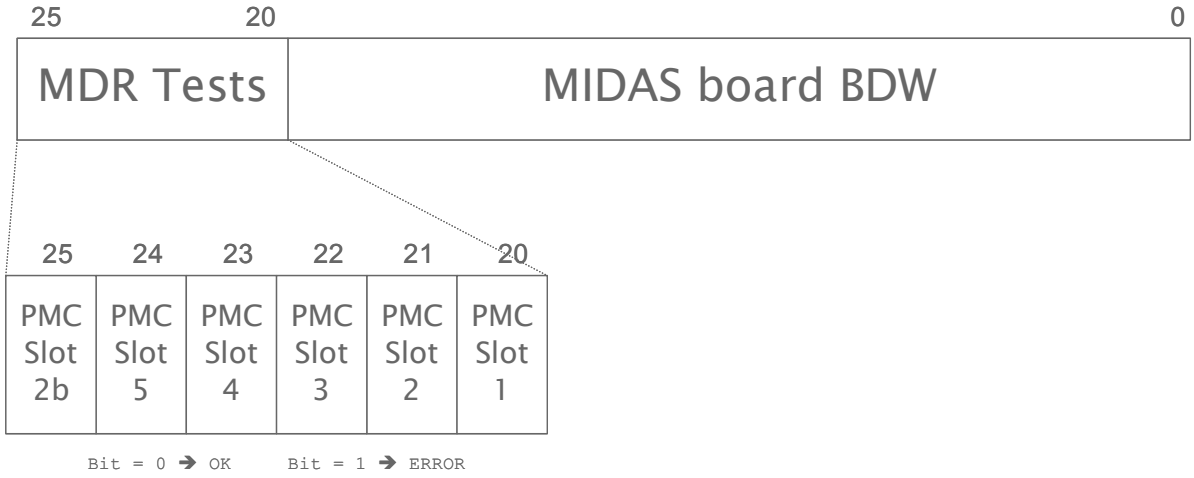
Boot Diagnostic Word

The result of each of these tests is stored on the board (in the i960 processor) and can be retrieved via the VMEbus at any time or by a program using the MDR API. The layout of the result is a word called the Boot Diagnostic Word (BDW).

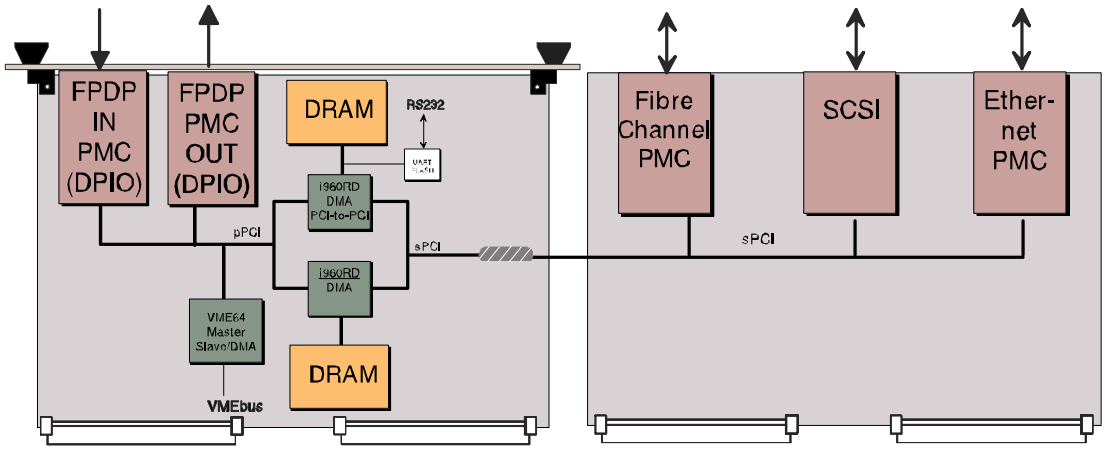
The layout of the BDW is the following:



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Example: MDR-250-135SE



Boot Diagnostic Word:

```
0000 0000 0000 0000 0100 0001 0000 0000
   0    0    0    0    4    1    0    0
```

Reading back the MDR Server Boot Status

There are two different options:

- Using a VME host, read the BDW via the VMEbus
- Using an application based on the MDR API, read the BDW using the `GetServerStatus` function

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In order to read the BDW via VMEbus, it is important to understand how this word is mapped to the VMEbus by the MDR Server:

- MDR with processor number 0: the BDW is mapped at the VME address `0x18`
- MDR with processor number greater than 0: the BDW is mapped at the VME address `0x400000 + ProcessorNumber*0x10000 + 0x18`

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

G Glossary

BSP	Board Support Package
DPIO	Digital Parallel Input Output
FC-AL	Fibre Channel Arbitrated Loop. A fast serial bus interface standard.
FPDP	Front Panel Data Port
HSSDC	High Speed Serial Data Connector
IP	Internet Protocol
JBOD	Just a Bunch Of Disks.
MDR	MIDAS Data Recorder. A high speed data recording system running on a MIDAS board.
MDR Client	A program (like the MDR Shell program) that controls the MDR by sending commands to the MDR Server.
MDR File System	File system used to organize all non-volatile MDR data, such as recordings, recording table and administration structures.
MDR Server	The program running on the i960 CPU on the MIDAS board that administers the data recorder devices and serves commands from MDR clients.
MDR Shell	The user interface providing the commands needed for MDR operation (mdrshell.exe on Windows and mdrshell on Solaris).
MIDAS	VME board with one or two i960 CPUs, and up to 5 PMC slots.
MTBF	Mean Time Between Failures
PCI	Peripheral Component Interconnect
PMC	PCI Mezzanine Card
RAID	Redundant Arrays of Inexpensive Disks
Recorder Block	Data block of size Recorder Block Size. The space available to the user to store data (on the storage device) is divided into a number of blocks called recorder blocks. The number of recorder blocks can be retrieved using the MDR Shell or MDR API.
Recorder Block Size	Block size of the MDR. This size can be retrieved using the MDR Shell or MDR API.
Recording	A continuous set of data stored in sequential recorder blocks on the storage device. More information about the recording attributes can be found in the header files..
Recording Table	A table used to store the identifiers, size and location of all recordings in the MDR system.
SBC	Single Board Computer. Typically a MIDAS-xx0 board.
Storage Device	The device which is used for storing the MDR recordings and MDR File System.
UserSpace	The part of the Storage Device that is available for the user.
UserSpaceSize	Size of UserSpace.
RecordingBlocks	Together with RecordingBlockSize, RecordingBlocks specify the size of each recording.
RecordingBlockSize	Size of a recording block.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group

Artisan Technology Group is your source for quality new and certified-used/pre-owned equipment

- FAST SHIPPING AND DELIVERY
- TENS OF THOUSANDS OF IN-STOCK ITEMS
- EQUIPMENT DEMOS
- HUNDREDS OF MANUFACTURERS SUPPORTED
- LEASING/MONTHLY RENTALS
- ITAR CERTIFIED SECURE ASSET SOLUTIONS

SERVICE CENTER REPAIRS
Experienced engineers and technicians on staff at our full-service, in-house repair center

WE BUY USED EQUIPMENT
Sell your excess, underutilized, and idle used equipment
We also offer credit for buy-backs and trade-ins
www.artisantg.com/WeBuyEquipment ↗

InstraView™ **REMOTE INSPECTION**
Remotely inspect equipment before purchasing with our interactive website at ***www.instraview.com*** ↗

LOOKING FOR MORE INFORMATION?
Visit us on the web at ***www.artisantg.com*** ↗ for more information on price quotations, drivers, technical specifications, manuals, and documentation

Contact us: (888) 88-SOURCE | sales@artisantg.com | www.artisantg.com